VOICE RECOGNITION PADA SISTEM KONTROL PINTU GARASI PINTAR MENGGUNAKAN *SUPPORT VECTOR MACHINE*

SKRIPSI

Oleh :
DIAH AYU RAHMA
NIM. 220605110006

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025

VOICE RECOGNITION PADA SISTEM KONTROL PINTU GARASI PINTAR MENGGUNAKAN *SUPPORT VECTOR MACHINE*

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
DIAH AYU RAHMA
NIM. 220605110006

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025

HALAMAN PERSETUJUAN

VOICE RECOGNITION* PADA SISTEM KONTROL PINTU GARASI PINTAR MENGGUNAKAN *SUPPORT VECTOR MACHINE

SKRIPSI

Oleh :
DIAH AYU RAHMA
NIM. 220605110006

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 8 Desember 2025

Pembimbing I,	Pembimbing II,
Ajib Hanani, M.T	Roro Inda Melani, M.T, M.Sc
NIP. 198407312023211013	NIP. 19780925 200501 2 008

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Supriyono, M. Kom
NIP. 19841010 201903 1 012

HALAMAN PENGESAHAN

VOICE RECOGNITION PADA SISTEM KONTROL PINTU GARASI PINTAR MENGGUNAKAN _SUPPORT VECTOR MACHINE_

SKRIPSI

Oleh :
DIAH AYU RAHMA
NIM. 220605110006

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 16 Desember 2025

Susunan Dewan Penguji

Ketua Penguji	: Johan Ericka Wahyu Prakasa, M.Kom NIP. 19831213 201903 1 004	( )
Anggota Penguji I	: Shoffin Nahwa Utama, M.T NIP. 198607032020121003	( )
Anggota Penguji II	: Ajib Hanani, M.T NIP. 198407312023211013	( )
Anggota Penguji III	: Roro Inda Melani, M.T, M.Sc NIP. 19780925 200501 2 008	( )

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Supriyono, M. Kom
NIP. 19841010 201903 1 012

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	: Diah Ayu Rahma
NIM	: 220605110006
Fakultas / Jurusan	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	: *Voice Recognition* Pada Sistem Kontrol Pintu Garasi Pintar Menggunakan *Support Vector Machine*

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 24 Desember 2025
Yang membuat pernyataan,

Diah Ayu Rahma
NIM.220605110006

MOTTO

"Kalau hidup sering 404 atau 500, mungkin itu bagian dari sistem update."

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh.

Syukur Alhamdulillah senantiasa penulis panjatkan kepada Allah SWT karena atas rahmat dan hidayah-Nya, sehingga penulis dapat tuntas menulis skripsi yang berjudul "*Voice Recognition* Pada Sistem Kontrol Pintu Garasi Pintar Menggunakan *Support Vector Machine*" dengan baik dan lancar.

Penulisan skripsi ini tidak terlepas dari bimbingan, doa, dan dukungan dari berbagai pihak. Oleh sebab itu, penulis menyampaikan terimakasih kepada :

1. Prof. Dr. Ilfi Nur Diana, M.Si., CAHRM., CRMP., selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. Agus Mulyono, M.Kes., selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Bapak Supriyono, M.Kom., selaku Ketua Program Studi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang
4. Bapak Ajib Hanani, M.T, selaku dosen pembimbing I yang senantiasa mendukung, membimbing, dan memberikan masukan dengan sabar selama penulisan.
5. Ibu Roro Inda Melani, M.T, M.Sc, selaku dosen pembimbing II yang telah memberikan masukan dan bimbingan untuk penulis dalam menyelesaikan skripsi ini.
6. Bapak Johan Ericka Wahyu Prakasa, M.Kom, selaku dosen penguji I dan Bapak Shoffin Nahwa Utama, M.T, selaku dosen penguji II yang telah

memberikan saran, masukan, kritik, dan diskusi yang membangun sehingga penulis dapat menyelesaikan skripsi dengan baik.

7. Nia Faricha S, Si selaku admin Program Studi Teknik Informatika yang selalu sabar memberikan informasi, membantu, dan memberikan arahan selama perkuliahan dan proses penulisan skripsi ini.
8. Dosen, laboran, dan jajaran staf Program Studi Teknik Informatika yang telah memberikan ilmu, pengetahuan, dan dukungan selama penulis menjalani studi.
9. Diri saya sendiri yang telah melewati segala tantangan, kegagalan, dan kegelisahan karena menjadi pemeran utama dalam perjalanan ini yang tetap bertahan dan terus berjuang meskipun jalan yang dilalui tak selalu dimudahkan.
10. Orangtua dan adik yang selalu menyertakan doa, dukungan, dan semangat kepada penulis selama masa perkuliahan dan penyusunan skripsi ini.
11. Teman terdekat, Fina Maslahatul Firhah yang saling mendukung dan menguatkan serta mendorong penulis untuk terus semangat dan berjuang bersama.
12. Sahabat sejak SMP, Rachma Pratama yang selalu menjadi pendengar setia dan memberikan semangat agar penulis terus maju dan tidak menyerah.
13. Keluarga besar “IT Dasaria“, Riza, Aulia, Satrio, Awanda, Indra, Adit, yang tidak pernah berhenti menyemangati penulis untuk segera menyelesaikan studi.
14. Taufik Ardiansyah Putra yang selalu memberikan motivasi, semangat, serta menemani penulis dari awal hingga akhir proses penyusunan skripsi ini.
15. Pooky dan Locky, kucing kesayangan yang setia menemani penulis selama proses penulisan skripsi.

Sebagai penutup, penulis menyadari bahwa skripsi ini bukanlah hasil kerja keras penulis seorang diri, melainkan hasil dukungan, doa, dan bantuan dari banyak pihak. Semoga apa yang telah penulis kerjakan dapat bermanfaat bagi diri sendiri dan orang lain. Penulis berharap, segala usaha dan doa yang tercurah dalam proses ini dapat memberikan hasil yang maksimal dan menjadi amal jariyah bagi semua pihak yang terlibat. Akhir kata, penulis memohon maaf atas segala kekurangan yang mungkin terdapat dalam penyusunan skripsi ini. Semoga Allah swt senantiasa memberkahi setiap langkah kita dan memberikan kemudahan dalam setiap usaha yang kita jalani.

Wassalamu'alaikum warahmatullahi wabarakatuh.

Malang, 30 November 2025

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Rahma, Diah Ayu. 2025. ***Voice Recognition*** **Pada Sistem Kontrol Pintu Garasi Pintar Menggunakan *Support Vector Machine*.** Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Ajib Hanani, M.T (II) Roro Inda Melani, M.T, M.Sc.

.

Kata kunci: *Internet of Things* (IoT), *Mel Frequency Ceptral Coefficients* (MFCC), Pintu Garasi Pintar, *Support Vector Machine, Voice Recognition*.

Penelitian ini mengembangkan sistem pintu garasi pintar berbasis speaker-dependent voice recognition untuk meningkatkan keamanan dan efisiensi akses pada lingkungan rumah. Sistem dirancang untuk mengenali identitas pengguna dan perintah suara menggunakan algoritma *Support Vector Machine* (SVM) serta pencocokan pola kata dengan *Regular Expression* (Regex). Proses pengiriman rekaman suara dari aplikasi *mobile* ke *backend* dilakukan melalui protokol HTTP, kemudian hasil prediksi diteruskan secara real time ke perangkat ESP32 melalui MQTT untuk menggerakkan servo, LED, buzzer, dan sensor *infrared*. Dataset terdiri dari 540 rekaman yang mencakup tiga jenis ujaran, masing-masing diklasifikasikan ke dalam kelas Me dan Notme. Model SVM dilatih menggunakan tiga skenario pembagian data, yaitu 90:10, 80:20, dan 70:30. Hasil pengujian menunjukkan bahwa model Support Vector Machine (SVM) dengan kernel Radial Basis Function (RBF), nilai Cost (C) = 10, dan Gamma (γ) = 0,5 memberikan performa terbaik pada skenario 70:30, dengan tingkat akurasi sebesar 97,53%. Selain itu, nilai Precision, Recall, dan F1-Score masing-masing mencapai 98%, yang menunjukkan kemampuan generalisasi model yang kuat. Sistem berhasil mengenali perintah "buka" dan "tutup" serta mengeksekusinya secara responsif, meskipun masih ditemukan error sistem sebesar 13,57% yang dipengaruhi oleh variasi kalimat, kesalahan transkripsi, serta kemiripan karakteristik akustik antarpenutur. Hasil penelitian menunjukkan bahwa pendekatan speaker-dependent berbasis SVM mampu memberikan performa tinggi untuk autentikasi suara dan kontrol perangkat pintar, serta memiliki potensi pengembangan lebih lanjut melalui perluasan dataset dan penyempurnaan modul pengenalan perintah.

ABSTRACT

Rahma, Diah Ayu. 2025. ***Voice Recognition for Smart Garage Door Control System Using Support Vector Machine*.** Undergraduate Thesis. Department of Informatics Engineering, Faculty of Science and Technology, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Advisors: (I) Ajib Hanani, M.T., (II) Roro Inda Melani, M.T., M.Sc.

This research develops a smart garage door system based on speaker-dependent voice recognition to enhance security and access efficiency in residential environments. The system is designed to identify users and interpret voice commands using the Support Vector Machine (SVM) algorithm combined with pattern matching through Regular Expressions (Regex). Audio recordings are transmitted from the mobile application to the backend via the HTTP protocol, and the prediction results are forwarded in real time to the ESP32 through MQTT to control the servo motor, LEDs, buzzer, and infrared sensor. The dataset consists of 540 audio recordings covering three types of utterances, each classified into Me and Notme categories. The SVM model was trained using three data-splitting scenarios: 90:10, 80:20, and 70:30. The experimental results indicate that the Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel, Cost (C) = 10, and Gamma (γ) = 0.5 achieved the best performance in the 70:30 scenario, with an accuracy of 97.53%. In addition, the Precision, Recall, and F1-score values each reached 98%, demonstrating strong generalization capability of the model. The system successfully recognized the "open" and "close" commands and executed them responsively, although a system error rate of 13.57% was still observed, mainly influenced by sentence variations, transcription errors, and acoustic similarities between speakers. These results indicate that the speaker-dependent SVM-based approach provides high performance for voice authentication and smart device control and has strong potential for further development through dataset expansion and refinement of the command recognition module.

Key words: *Internet of Things (IoT), Mel Frequency Ceptral Coefficients (MFCC), Smart Door Garage, Support Vector Machine, Voice Recognition.*

مستخلص البحث

رحمة، دياه أيو. 2025. **التعرف على الصوت في نظام التحكم في باب الجراج الذكي باستخدام آلة المتجه الداعم**. البحث الجامعي. قسم الهندسة المعلوماتية، كلية العلوم والتكنولوجيا بجامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرف الأول: عجيب حناني، الماجستير؛ المشرف الثاني: رورو إنده ميلاني، الماجستير.

تطوير هذا البحث لنظام باب الجراج الذكي القائم على التعرف على الصوت المعتمد على المتحدث يهدف إلى تعزيز الأمان وكفاءة الوصول في بيئة المنزل. تم تصميم النظام للتعرف على هوية المستخدم وأوامر الصوت باستخدام خوارزمية آلة المتجهات الداعمة (SVM) بالإضافة إلى مطابقة نمط الكلمات باستخدام التعبيرات العادية (Regex). تتم عملية إرسال تسجيلات الصوت من تطبيق الهاتف المحمول إلى الخادم عبر بروتوكول HTTP، ثم يتم تمرير نتائج التنبؤ في الوقت الفعلي إلى جهاز ESP32 عبر بروتوكول MQTT لتشغيل السيرفو، LED، الجرس، ومستشعر الأشعة تحت الحمراء. تتكون مجموعة البيانات من 540 تسجيلًا تشمل ثلاثة أنواع من الكلام، تم تصنيف كل منها إلى فئتين هما Me و Notme. تم تدريب نموذج SVM باستخدام ثلاث سيناريوهات لتقسيم البيانات، وهي 90:10 و80:20 و70:30. أظهرت نتائج الاختبارات أن نموذج آلة المتجهات الداعمة (SVM) باستخدام نواة Radial Basis Function (RBF) وقيمة Cost (C) = 10 وقيمة Gamma (γ) = 0.5 حقق أفضل أداء في سيناريو 70:30، حيث بلغت دقة التصنيف 97.53٪، كما بلغت قيم Precision وRecall و F1 نسبة 98٪. نجح النظام في التعرف على أوامر «افتح» و«أغلق» وتنفيذها بسرعة، على الرغم من وجود خطأ في النظام بنسبة 13.57٪، والذي تأثر بتنوع صياغة الجمل، وأخطاء النسخ، والتشابه في الخصائص الصوتية بين المتحدثين. وتشير نتائج البحث إلى أن المدخل المعتمد على المتحدث باستخدام SVM قادر على تقديم أداء عالٍ في مصادقة الصوت والتحكم في الأجهزة الذكية، كما يمتلك إمكانية للتطوير المستقبلي من خلال توسيع مجموعة البيانات وتحسين وحدة التعرف على الأوامر.

الكلمات الرئيسية: التعرف على الصوت، آلة المتجهات الداعمة (SVM)، إنترنت الأشياء (IoT)، معاملات ميل-تردد سبسترال (MFCC)، باب الجراج الذكي.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi di era digital telah mendorong penerapan konsep *smart home* sebagai upaya meningkatkan kenyamanan, efisiensi, dan keamanan bagi pengguna. Salah satu implementasi yang cukup penting adalah sistem garasi otomatis. Banyak pintu garasi masih dioperasikan secara manual sehingga memerlukan tenaga tambahan serta tidak praktis ketika kondisi cuaca kurang mendukung atau ketika pengguna sedang terburu-buru. Kebutuhan akan sistem yang lebih mudah, aman, dan efisien memunculkan urgensi pengembangan teknologi garasi otomatis berbasis kontrol suara. Berbagai penelitian sebelumnya telah merancang sistem garasi otomatis dengan pendekatan berbeda. Ermawati dkk (2023) menggunakan modul voice recognition berbasis *keyword spotting* untuk mengendalikan motor *stepper*, tetapi masih terbatas pada pengenalan kata tertentu tanpa kemampuan memahami variasi ucapan pengguna.

Teknologi *voice recognition* konvensional memiliki sejumlah keterbatasan, khususnya dalam hal akurasi dan kemampuan beradaptasi terhadap perbedaan intonasi, artikulasi, dan pola bicara setiap pengguna. Penggunaan metode *machine learning* menawarkan peningkatan performa karena memiliki kemampuan mengenali pola suara secara lebih fleksibel. Beberapa studi menunjukkan bahwa SVM efektif digunakan pada sistem pengenalan suara karena dapat membedakan karakteristik suara secara presisi (Nang An et al., 2023). Integrasi SVM dengan teknologi *speech-to-text* (STT) memberi kemampuan bagi sistem untuk memahami

perintah dalam bentuk kata maupun kalimat sebelum memetakan hasilnya menjadi instruksi operasional untuk garasi.

Urgensi penelitian ini diperkuat oleh prinsip amanah dalam Islam. *QS. An-Nisa [4]:58* menegaskan pentingnya menunaikan amanah dan melaksanakan tanggung jawab secara adil.

اِنَّ اللّٰهَ يَأْمُرُكُمْ اَنْ تُؤَدُّوا الْاَمٰنٰتِ اِلٰٓى اَهْلِهَاۙ وَاِذَا حَكَمْتُمْ بَيْنَ النَّاسِ اَنْ تَحْكُمُوْا بِالْعَدْلِۗ اِنَّ اللّٰهَ نِعِمَّا يَعِظُكُمْ بِهٖۗ اِنَّ اللّٰهَ كَانَ سَمِيْعًاۢ بَصِيْرًا

"Sesungguhnya Allah menyuruh kamu menyampaikan amanah kepada pemiliknya. Apabila kamu menetapkan hukum di antara manusia, hendaklah kamu tetapkan secara adil. Sesungguhnya Allah memberi pengajaran yang paling baik kepadamu. Sesungguhnya Allah Maha Mendengar lagi Maha Melihat.".

Menurut tafsir tahlili, ayat ini menegaskan kewajiban menunaikan amanah kepada pihak yang berhak serta menegakkan hukum dengan adil. Makna amanah mencakup amanah manusia kepada Allah berupa ketaatan, amanah antar sesama berupa menjaga titipan, menepati janji, serta amanah terhadap diri sendiri untuk tidak melakukan hal-hal yang merugikan dunia maupun akhirat. Dalam konteks penelitian ini, prinsip amanah dapat dimaknai sebagai kewajiban menjaga dan melindungi harta benda, termasuk kendaraan yang tersimpan di garasi. Dengan demikian, pengembangan sistem garasi pintar berbasis teknologi tidak hanya relevan dari sisi efisiensi, tetapi juga sejalan dengan ajaran Islam yang menekankan pentingnya keamanan dan tanggung jawab dalam kehidupan sehari-hari.

Pada *QS. Ali Imran [3]:191* juga menyatakan:

الَّذِيْنَ يَذْكُرُوْنَ اللّٰهَ قِيَامًا وَّقُعُوْدًا وَّعَلٰى جُنُوْبِهِمْ وَيَتَفَكَّرُوْنَ فِيْ خَلْقِ السَّمٰوٰتِ وَالْاَرْضِۚ رَبَّنَا مَا خَلَقْتَ هٰذَا بَاطِلًاۚ سُبْحٰنَكَ فَقِنَا عَذَابَ النَّارِ

"(yaitu) orang-orang yang mengingat Allah sambil berdiri, duduk, atau dalam keadaan berbaring, dan memikirkan tentang penciptaan langit dan bumi (seraya berkata), "Ya Tuhan kami, tidaklah Engkau menciptakan semua ini sia-sia. Mahasuci Engkau. Lindungilah kami dari azab neraka.".

Menurut tafsir tahlili, ayat ini menggambarkan ciri khas *ulul albāb*, yakni orang-orang berakal yang selalu mengambil manfaat dari ciptaan Allah. Mereka senantiasa berdzikir dalam setiap keadaan berdiri, duduk, maupun berbaring seraya memikirkan keajaiban penciptaan langit dan bumi. *Tafakkur* ini melahirkan kesadaran akan kebesaran Allah, sehingga manusia menyadari bahwa tidak ada satu pun ciptaan Allah yang sia-sia. Fenomena kompleks seperti penciptaan langit dan bumi, silih bergantinya malam dan siang hingga keteraturan tata surya, semuanya menunjukkan kesempurnaan kekuasaan Allah. Hal ini mendorong manusia untuk senantiasa bersyukur, mengambil hikmah, dan memanfaatkan ciptaan Allah secara tepat demi kemaslahatan hidup.

Asbari et al. (2025) dalam *Journal of Information Systems and Management* menekankan bahwa akal sehat merupakan instrumen penting bagi manusia untuk merespons perkembangan teknologi di era digital. Ayat tentang *ulul albāb* tidak hanya relevan dalam konteks *tafakkur* atas ciptaan Allah, tetapi juga dapat dimaknai sebagai dorongan agar manusia menjaga rasionalitas, berpikir kritis, serta memanfaatkan teknologi secara sehat dan produktif. Dengan demikian, pemanfaatan akal melalui pengembangan ilmu pengetahuan dan teknologi, termasuk sistem garasi otomatis berbasis suara, merupakan bentuk aktualisasi karunia Allah agar ciptaan-Nya membawa manfaat nyata bagi kehidupan Dengan menggabungkan prinsip amanah (*QS. An-Nisa [4]:58*) dan prinsip bahwa ciptaan Allah memiliki manfaat dan tidak diciptakan dengan sia-sia (*QS. Ali Imran*

[3]:191), penelitian ini menegaskan bahwa inovasi teknologi, termasuk sistem garasi otomatis berbasis suara, bukan hanya relevan dari sisi teknis dan sosial, tetapi juga memiliki landasan moral dan agama. Dengan demikian, penelitian ini diharapkan mampu memberikan kontribusi nyata dalam meningkatkan keamanan, kenyamanan, dan efisiensi bagi pengguna, sekaligus selaras dengan ajaran Islam.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka rumusan masalah dalam penelitian ini adalah bagaimana mengukur performa sistem pintu garasi pintar berbasis *speaker-dependent voice recognition* menggunakan algoritma *Support Vector Machine*?

1.3 Batasan Masalah

Penelitian ini memiliki beberapa Batasan masalah di antaranya:

1. Penelitian ini dibatasi pada penggunaan bahasa Indonesia sebagai bahasa utama dalam pengenalan dan pemrosesan perintah suara. Sistem tidak dirancang untuk mengenali perintah dalam bahasa lain di luar bahasa Indonesia.
2. Data penelitian ini diperoleh melalui proses perekaman suara secara mandiri dari total 19 partisipan, yang terdiri atas 1 pengguna sah (*Me*) dan 18 penutur lain (*Notme*). Seluruh perekaman dilakukan menggunakan mikrofon eksternal pada jarak sekitar ±20 cm dari mulut di ruangan dengan tingkat kebisingan rendah hingga sedang.

3. Total data mentah yang digunakan dalam penelitian ini berjumlah 540 berkas audio, yang terdiri atas tiga jenis ujaran perintah “Buka”, “Tutup”, dan *filler.*

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk mengukur performa sistem garasi pintar berbasis *speaker-dependent voice recognition* menggunakan algoritma *Support Vector Machine*.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat dalam beberapa aspek berikut:

1. Menjadi kontribusi dalam bidang ilmu pengetahuan dan teknologi, khususnya dalam pengembangan sistem pengenalan suara berbasis machine learning dengan pendekatan algoritma *Support Vector Machine* untuk aplikasi asistif.
2. Memberikan solusi yang dapat diimplementasikan pada sistem Pintu Garasi Pintar sehingga pengguna dapat mengakses dan mengendalikan pintu garasi secara lebih mudah, aman, dan efisien.

BAB II
STUDI PUSTAKA

2.1 Penelitian Terkait

Penelitian mengenai garasi otomatis telah dilakukan dengan berbagai pendekatan teknologi. Prayetno (2022) merancang *Prototype Pintu Garasi Otomatis Menggunakan Sensor Suara Berbasis Arduino UNO* dengan memanfaatkan sensor suara FC-04 sebagai penerima perintah. Sistem bekerja optimal pada frekuensi 3020 Hz (76 dB) dan 3150 Hz (79 dB) dengan tingkat keberhasilan 100%. Tingkat responsivitasnya tinggi, tetapi mekanisme pendeteksian masih terbatas pada pengenalan suara berbasis frekuensi sehingga aspek keamanannya belum memadai. Mohamed Ariff dkk. (2022) melalui penelitian berjudul *Design and Development of a Smart Garage Door System* mengembangkan prototipe garasi otomatis berbasis IoT menggunakan Arduino, aplikasi Blynk, dan integrasi Google Assistant. Sistem memungkinkan kontrol pintu garasi melalui *smartphone* maupun perintah suara jarak jauh dan mampu beroperasi dengan baik, meskipun ketergantungan pada platform pihak ketiga membuat fleksibilitas dan kemandiriannya terbatas.

Pengembangan sistem *voice recognition* pada lingkungan *smart home* juga menunjukkan variasi hasil. Ramba dan Aria (2020) merancang sistem *home automation* dengan kendali suara berbasis *Convolutional Neural Network* (CNN) untuk mengontrol perangkat rumah tangga seperti lampu, pintu, dan kipas angin. Sistem mencapai akurasi 100% pada kondisi ruangan senyap (24 dB), 67,67% pada kondisi bising 42 dB, dan 51,67% pada 52 dB. Temuan tersebut menegaskan

bahwa CNN mampu mengenali suara dengan baik, meskipun performanya menurun signifikan pada lingkungan dengan tingkat kebisingan tinggi. Irugalbandara dkk. (2023) mengembangkan sistem *smart home* menggunakan *on-device speech recognition* yang mencakup VAD, *wake word detection*, CNN-based STT dengan CTC *decoder*, serta BERT-*based* NLU. Sistem dapat berjalan tanpa koneksi internet, memberikan respons cepat, meningkatkan keamanan terhadap serangan siber, dan menyediakan fitur monitoring konsumsi daya. Kelemahannya terletak pada kebutuhan peningkatan akurasi, perbaikan antarmuka pengguna, dan penguatan keamanan jaringan Wi-Fi *mesh*.

Upaya peningkatan kinerja sistem garasi berbasis suara juga dilakukan oleh Ermawati dkk. (2023) yang merancang prototipe pengendali pintu garasi menggunakan Arduino dan motor *stepper* Nema 17, dilengkapi dengan *voice recognition* dan kontrol PID. Sistem mampu merespons perintah suara seperti penuh, setengah, dan tertutup dengan jeda sekitar 1,5 detik. Kinerja tersebut cukup baik, tetapi cakupan penggunaan masih terbatas pada satu pengguna dan jarak tertentu.

Rangkaian penelitian di bidang ini menunjukkan bahwa sistem garasi otomatis masih banyak mengandalkan sensor suara konvensional atau modul sederhana yang hanya mendukung perintah dasar. Penelitian lain pada ranah *smart home* membuka peluang integrasi teknologi *speech-to-text* dan *machine learning* untuk meningkatkan fleksibilitas interaksi pengguna. Penggabungan SVM sebagai mekanisme otentikasi suara dan Regex sebagai pencocokan perintah menawarkan

pendekatan yang lebih aman, adaptif, dan relevan bagi kebutuhan pengguna modern.

Tabel 2.1. Penelitian Terkait

No.	Peneliti	Judul	Metode	Hasil
1.	Chandra Irugalbandara, et al (2023)	*A Secure and Smart Home Automation System with Speech Recognition and Power Measurement Capabilities*	*On-device speech recognition* (VAD, *wake word*, CNN-*based* STT + CTC *decoder*, BERT-*based* NLU)	Sistem mampu berfungsi tanpa internet, memberikan respon cepat, aman dari serangan siber, serta dilengkapi fitur tambahan seperti monitoring konsumsi daya dan optimasi perangkat. Namun, masih perlu peningkatan akurasi, UI, dan keamanan Wi-Fi *mesh*.
2.	Lery Sakti Aria & Muhammad Aria Rajasa Pohan (2020)	Perancangan Sistem *Home Automation* Dengan Kendali Perintah Suara Menggunakan *Deep Learning Convolutional Neural Network* (DL-CNN)	Deep Learning - CNN	Sistem mampu mengendalikan perangkat rumah (lampu, pintu, kipas) dengan akurasi 100% di ruangan senyap (24 dB), 67,67% pada noise 42 dB, dan 51,67% pada *noise* 52 dB. Lebih optimal digunakan pada ruangan dengan kebisingan rendah.
3.	Mohamed Ariff & Mohamed Imran (2022)	*Design and Development of a Smart Garage Door System*	Arduino + IoT (aplikasi Blynk & Google Assistant)	Sistem berhasil melakukan operasi dasar buka-tutup garasi secara jarak jauh melalui aplikasi *mobile* dan perintah suara Google Assistant. Memberikan

				kemudahan dan efisiensi, meskipun masih ada keterbatasan pada aspek keamanan dan ketergantungan pada platform pihak ketiga.
4.	Mochamad Ady Prayetno (2022)	*Prototype* Pintu Garasi Otomatis Menggunakan Sensor Suara Berbasis Arduino UNO	Arduino UNO dengan sensor suara FC-04	Sistem pintu garasi otomatis bekerja responsif dengan tingkat keberhasilan 100% pada frekuensi 3020 Hz (76 dB) dan 3150 Hz (79 dB)
5.	Yuli Ermawati, et al (2023)	Prototype Pengontrol Pintu Garasi Rumah Dengan Motor *Stepper* Berbasis Arduino Menggunakan Perintah Suara	Arduino, sensor *voice recognition*, motor *stepper* Nema 17 dengan kontrol PID	Sistem berhasil merespons perintah suara (*penuh, setengah, tertutup*), dengan jeda optimal 1,5 detik untuk akurasi respon. Efektif namun masih terbatas pada satu pengguna dan jarak tertentu.

2.2 Teori Suara

Suara merupakan getaran mekanik yang merambat melalui medium udara dan dapat didengar oleh telinga manusia. Getaran ini menyebabkan perubahan tekanan udara secara periodik yang membentuk gelombang longitudinal. Menurut Adler et al. (2013), suara dihasilkan oleh pita suara dengan kerja sama seluruh organ penghasil suara, di mana setiap individu memiliki karakteristik dan tingkat frekuensi yang berbeda-beda. Darmawan (2017) menambahkan bahwa suara dihasilkan oleh getaran yang terjadi tiap detik pada suatu benda yang merambat

melalui udara dan dapat diterima baik oleh telinga manusia maupun alat perekam seperti mikrofon.

Secara fisik, suara memiliki tiga unsur utama yaitu frekuensi, amplitudo, dan timbre. Frekuensi adalah jumlah getaran per detik yang menentukan tinggi rendahnya nada dan diukur dalam satuan Hertz (Hz). Amplitudo menunjukkan besar kecilnya simpangan getaran yang berpengaruh terhadap keras lembutnya suara, sedangkan timbre atau warna suara merupakan kualitas khas dari suatu bunyi yang membedakannya dari bunyi lain meskipun memiliki frekuensi dan amplitudo yang sama.

Rentang frekuensi pendengaran manusia umumnya berada antara 20 Hz hingga 20.000 Hz, di mana suara di bawah 20 Hz disebut infrasonik dan di atas 20 kHz disebut ultrasonik. Rentang ini dapat bervariasi tergantung usia dan kondisi fisiologis seseorang. Dalam *Sound Film Book*, Darmawan (2017) membagi wilayah frekuensi suara ke dalam oktaf dengan perbandingan 2:1 antar frekuensi, yang mencakup hampir sepuluh oktaf pada pendengaran manusia. Wilayah tersebut dibagi menjadi lima bagian, yaitu *Low Bass* (20–80 Hz) yang memberikan kesan berat, *Upper Bass* (80–320 Hz) yang menambah kekuatan suara, *Midrange* (320–2.560 Hz) yang menjadi wilayah utama suara manusia, *Upper Midrange* (2.560–5.120 Hz) yang menentukan kejernihan dan artikulasi, serta *Treble* (5.120–20.480 Hz**)** yang memberi kesan terang dan detail pada suara.

2.3 *Speaker Dependent*

Speaker-dependent adalah pendekatan pengenalan suara yang dilatih secara khusus menggunakan data dari satu pengguna tertentu, sehingga performa sistem

menjadi sangat optimal untuk pengguna yang sama. Marini et al., (2021)menjelaskan bahwa dalam *Automatic Speech Recognition* (ASR), model *speaker-dependent* memberikan keuntungan karena parameter ekstraksi fitur seperti ukuran jendela (*window size*) dan *frame shift* dapat disesuaikan secara spesifik dengan karakteristik suara pengguna. Penyesuaian ini memungkinkan sistem menangkap pola vokal yang lebih konsisten, sehingga mengurangi tingkat kesalahan pengenalan.

2.4 *Voice Recognition*

Voice recognition adalah teknologi yang memungkinkan komputer atau perangkat digital mengenali kata-kata yang diucapkan dengan cara mengubah sinyal suara menjadi representasi digital, kemudian mencocokkannya dengan pola suara yang telah disimpan. *Voice recognition* bekerja melalui proses digitalisasi sinyal audio, ekstraksi ciri, dan pencocokan pola untuk mengidentifikasi makna atau perintah yang diucapkan pengguna.

2.5 *Mel-Frequency Cepstral Coefficients* (MFCC)

MFCC merupakan representasi fitur audio yang mengekstrak pola energi pada amplop spektral suara atau "sidik jari" karakteristik vokal. Proses ini dilakukan dengan memetakan spektrum ke skala Mel dan mengompresi hasilnya menggunakan DCT sehingga koefisien yang dihasilkan lebih ringkas dan tidak saling berkorelasi. MFCC selaras dengan cara telinga manusia membedakan energi antar-pita frekuensi sehingga metode ini banyak digunakan pada sistem pengenalan suara, verifikasi penutur, dan analisis audio.

Tahap awal dimulai dari *waveform* berupa sinyal audio mentah $x[n]$ (mono) yang berisi amplitudo terhadap waktu. Sinyal ini kemudian dinormalisasi untuk menambah keseragaman data, termasuk pengonversian seluruh rekaman ke format *wav* 16 kHz, penyelarasan level volume, serta penyeragaman durasi. *Pre-emphasis* diterapkan melalui rumus $y[n] = x[n] - \alpha x[n-1]$ (α sekitar 0,95–0,98) yang menonjolkan komponen frekuensi tinggi dan membantu meningkatkan kejelasan informasi pada proses ekstraksi ciri.

Sifat sinyal wicara yang tidak stasioner membuatnya dibagi ke dalam *frame* pendek agar setiap potongan dapat dianggap kuasi-stasioner. *Frame* berukuran sekitar 20 ms digunakan untuk melihat perubahan energi secara lokal terhadap waktu. Spektrum frekuensi diperoleh menggunakan *Discrete Fourier Transform* (DFT) sehingga karakter energi dalam setiap *frame* dapat tergambarkan. Spektrum yang terbentuk kemudian diubah ke skala logaritmik untuk menyesuaikan karakter persepsi pendengaran manusia yang lebih sensitif pada perubahan energi di intensitas rendah.

Pemetaan spektrum log-energi ke skala Mel dilakukan menggunakan rangkaian filter segitiga (*Mel filterbank*). Energi pada tiap bin frekuensi dijumlahkan mengikuti pita-pita Mel sehingga menghasilkan representasi energi yang ringkas dan relevan secara perseptual. Skala Mel merapat di frekuensi rendah dan merenggang di frekuensi tinggi sehingga pola yang dihasilkan mencerminkan karakter pendengaran manusia. Vektor energi Mel yang terbentuk kemudian dikompresi dengan DCT untuk mengurangi korelasi antar-pita. Proses ini menghasilkan MFCC sebagaimana dijelaskan oleh Utama dkk. (2025).

2.6 Support Vector Machine (SVM)

Support Vector Machine (SVM) adalah metode klasifikasi biner yang diperkenalkan oleh Vapnik dan banyak digunakan dalam pengenalan suara karena akurasinya yang tinggi. Prinsip utama SVM adalah membangun *hyperplane* dengan margin maksimum yang dapat memisahkan dua kelas secara optimal, sehingga lebih tahan terhadap kesalahan klasifikasi akibat *outlier*. Untuk data yang tidak linier, SVM menggunakan fungsi kernel seperti Linear, *Polynomial*, *Radial Basis Function* (RBF), atau *Sigmoid* untuk mentransformasikan data ke ruang dimensi lebih tinggi agar dapat dipisahkan secara linier. Proses klasifikasi pada SVM dapat dituliskan dengan persamaan:

$$f(x) = sign \sum_{i=1}^{L} \alpha_i \, . \, l_i . K(x_i, x) \; + \; b \tag{2.1}$$

Dengan α_i sebagai *Lagrange multipliers*, l_i label kelas (−1 atau +1), *K* fungsi kernel, dan *b* nilai bias. Hanya sebagian data latih yang digunakan dalam model, yaitu *support vector* (data dengan $\alpha_i > 0$). Handoko & Suyanto (2019) menunjukkan bahwa kombinasi SVM dengan ekstraksi ciri suara menggunakan MFCC mampu mengklasifikasikan gender dengan akurasi tinggi. Kernel *Polynomial* dengan *degree* = 1 memberikan akurasi terbaik mencapai 99,68% pada data bersih, meskipun akurasi menurun signifikan ketika data uji diberi *noise*, menandakan keterbatasan SVM dalam kondisi *noise* tinggi.

2.7 *Regular Expression* (Regex)

Regular Expression (Regex) merupakan algoritma pencocokan pola (*pattern matching*) yang digunakan untuk menemukan atau memvalidasi teks

tertentu dalam sebuah *string*. Menurut Bahar & Daniel Yc. Raban (2023), Regex dapat diaplikasikan dalam berbagai bidang, salah satunya untuk mendeteksi pola nominal uang kertas Rupiah. Pada penelitian tersebut, kamera *smartphone* digunakan untuk mengambil gambar uang kertas, kemudian teks nominal diekstraksi menggunakan *Optical Character Recognition* (OCR). Hasil teks ini selanjutnya diproses oleh Regex untuk mencari pola yang sesuai dengan nilai nominal. Jika pola cocok ditemukan, sistem memberikan keluaran berupa teks nominal di layar, getaran khusus sesuai nilai uang, serta suara melalui fitur *Talkback*.

Regex bekerja dengan aturan pencocokan berbasis karakter, simbol, dan urutan tertentu sehingga mampu mengenali pola teks dengan cepat dan akurat. Keunggulan utamanya adalah fleksibilitas dalam mendeteksi variasi format teks yang berbeda. Uji coba pada 210 sampel uang kertas dengan melibatkan 10 responden tunanetra menunjukkan akurasi 100%, membuktikan bahwa kombinasi OCR dan Regex efektif dalam mengenali pola teks nominal. Temuan ini menegaskan bahwa Regex dapat menjadi metode pendukung yang ringan, efisien, dan andal dalam sistem pengenalan berbasis teks maupun suara.

2.8 Kata *Filler*

Dalam linguistik dan pemrosesan suara, *filler words* atau *kata bebas* merupakan bentuk ujaran non-leksikal yang sering muncul dalam percakapan spontan. Menurut Zhu et al. (2022), kata *filler* seperti “uh” atau “um” merupakan bunyi atau kata yang digunakan seseorang untuk menandakan jeda berpikir. Kata-kata ini tidak menambah makna semantik pada kalimat, tetapi berfungsi sebagai

penanda jeda atau refleksi spontan ketika penutur sedang menyusun pikiran atau mempertahankan giliran berbicara.

Dalam konteks pemrosesan suara otomatis (*Automatic Speech Recognition*/ASR), *filler* termasuk ke dalam kategori *speech disfluencies*, yaitu ketidakteraturan dalam ujaran seperti pengulangan, gagap, atau koreksi (Zhu et al., 2022). *Filler* dianggap sebagai suara non-leksikal yang tidak memiliki makna kontekstual, sehingga sering diabaikan atau dihapus dalam proses transkripsi otomatis. Namun, dalam sistem pengenalan perintah suara, keberadaan *filler* justru penting karena dapat digunakan untuk melatih sistem agar mampu membedakan antara ucapan bermakna (perintah) dan ucapan spontan yang tidak relevan.

2.9 *Hypertext Transfer Protocol* (HTTP)

Hypertext Transfer Protocol (HTTP) adalah sebuah protokol komunikasi yang digunakan untuk proses permintaan (*request*) dan jawaban (*response*) antara *client* dan *server* pada web. Pada mekanisme ini, *client* biasanya web browser memulai komunikasi dengan membuat koneksi TCP/IP ke server pada port tertentu. Setelah koneksi terbentuk, *client* mengirimkan kode permintaan, seperti "`GET / HTTP/1.1`", yang kemudian dapat disertai *header* dan *body* untuk menjelaskan detail permintaan tersebut. Server yang menerima permintaan tersebut akan memberikan kode balasan serta mengirimkan data atau halaman web yang diminta (Zabar & Novianto, 2015).

2.10 *Internet of Things*

Menurut Susanto et al. (2022) *Internet of Things* (IoT) merupakan teknologi yang memungkinkan berbagai objek saling terhubung melalui jaringan internet

sehingga memudahkan aktivitas sehari-hari serta meningkatkan efisiensi. IoT membuat perangkat dapat berinteraksi dan bertukar data secara otomatis, dan penerapannya kini semakin luas di berbagai bidang, baik skala kecil maupun besar. Teknologi ini juga mendorong munculnya peluang serta inovasi baru dalam pengembangan sistem modern.

Megawati (2021) menyebutkan bahwa IoT bekerja dengan memanfaatkan data digital yang dikumpulkan oleh berbagai sensor seperti RFID, inframerah (IR), atau GPS. Informasi yang diperoleh diproses lebih lanjut agar perangkat yang saling terhubung dapat beroperasi, berinteraksi, dan dikendalikan secara lebih efektif.

2.11 *Message Queuing Telemetry Transport (MQTT)*

MQTT adalah protokol komunikasi ringan berbasis *publish/subscribe* di atas TCP/IP yang dirancang untuk perangkat IoT berdaya rendah dan jaringan tak stabil/ber-latensi tinggi. MQTT bekerja dengan memakai *broker* sebagai perantara, *publisher* mengirim pesan ke sebuah topik, dan *subscriber* menerima pesan dengan cara berlangganan topik yang sama. Pendekatan ini membuat sistem longgar keterikatannya (*decoupled*) pengirim dan penerima tidak perlu saling tahu keberadaan atau status masing-masing, sehingga skalabel dan hemat sumber daya (Hanani & Hariyadi, 2020).

2.12 Arduino IDE

Arduino IDE adalah platform pemrograman berbasis C/C++ yang digunakan secara luas untuk pengembangan sistem berbasis mikrokontroler. Dalam penelitian oleh Hercog et al. (2023), Arduino IDE disebut sebagai salah satu alat edukatif dan *prototyping* utama dalam pengembangan perangkat ESP32 berbasis sensor.

2.13 ESP32

ESP32 adalah mikrokontroler yang memiliki prosesor *dual-core* dengan arsitektur Xtensa LX6 serta dilengkapi konektivitas Wi-Fi dan Bluetooth. Menurut (Hercog et al., 2023) dalam jurnal berjudul *Design and Implementation of ESP32-Based IoT Devices*, ESP32 banyak digunakan dalam sistem IoT karena memiliki kemampuan pemrosesan data cukup tinggi serta kompatibel dengan berbagai protokol komunikasi seperti UART, I2C, dan SPI. Keunggulannya terletak pada efisiensi konsumsi daya, biaya produksi yang rendah, dan fleksibilitas tinggi dalam pengembangan sistem terintegrasi.

2.14 *Infrared* Sensor

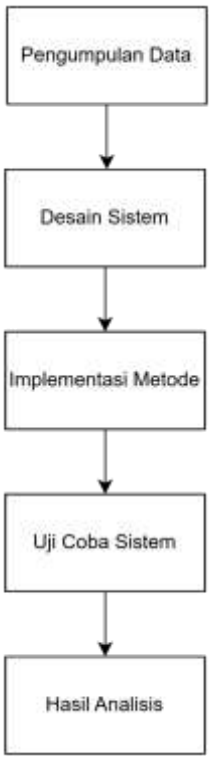
Sensor *infrared* (IR) adalah instrumen elektronik yang digunakan untuk mendeteksi karakteristik lingkungan di sekitarnya dengan mengirimkan atau menerima radiasi infra merah. Sensor ini bekerja dengan menangkap perubahan energi panas atau pantulan cahaya *infrared* dari suatu objek. Dalam sistem dasar, IR LED berfungsi sebagai pemancar, sementara *photodiode* atau fototransistor berfungsi sebagai penerima yang sensitif terhadap panjang gelombang IR. Ketika cahaya *infrared* mengenai penerima, perubahan resistansi atau tegangan akan muncul sesuai intensitas cahaya yang diterima, sehingga objek atau hambatan dapat dideteksi. Sensor IR umum digunakan untuk deteksi objek, pengukuran panas, pemantauan gerakan, serta berbagai aplikasi otomasi dan perangkat elektronik (Ajmera, 2017).

BAB III

DESAIN DAN IMPLEMENTASI

3.1 Desain Penelitian

Desain penelitian adalah kerangka kerja atau rencana yang digunakan untuk mengatur, menjalankan, dan mengarahkan sebuah penelitian yang mencangkup metode, prosedur, dan strategi yang digunakan peneliti untuk menjawab pertanyaan penelitian atau menguji hipotesis. Adapun jenis penelitian yang digunakan pada penelitian ini adalah penelitian kuantitatif, yang berarti analisis data yang digunakan dapat diukur. Alur penelitian terkait dapat dilihat melalui diagram berikut:



Gambar 3.1. Alur Penelitian

Pengumpulan data suara menjadi tahap awal dalam penelitian ini. Sampel direkam dari pengguna dengan dua instruksi utama, yaitu “Buka” dan “Tutup”. Rekaman dibuat dalam format *.wav* menggunakan mikrofon eksternal pada lingkungan *indoor* dengan tingkat kebisingan rendah hingga sedang. Dataset terdiri atas dua kelompok suara, yakni suara pemilik (kelas *Me*) dan suara non-pemilik (kelas *Notme*), untuk mendukung proses identifikasi *speaker* berbasis metode *speaker-dependent* dengan *voice recognition*.

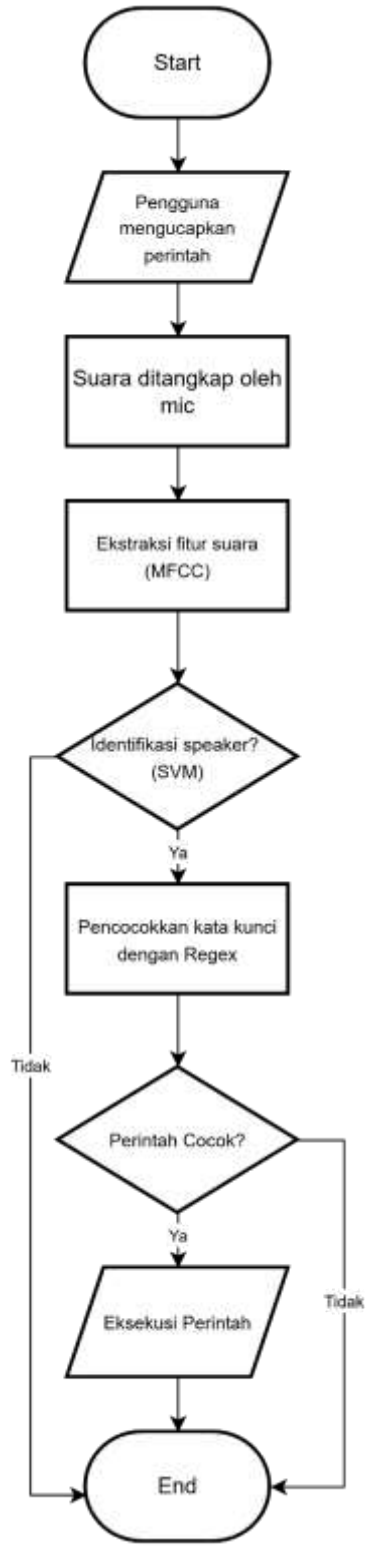
Desain sistem disusun setelah dataset terkumpul. Proses pemrosesan sinyal suara dirancang mulai dari ekstraksi ciri menggunakan MFCC hingga tahapan pengenalan. Hasil ekstraksi fitur diproses dalam dua bagian: identifikasi *speaker* menggunakan algoritma SVM untuk membedakan suara pemilik dan non-pemilik, serta pencocokan instruksi melalui Regex untuk mengenali kata kunci “Buka” dan “Tutup”. Implementasi metode dilakukan dengan mengintegrasikan seluruh *pipeline* pemrosesan suara ke dalam perangkat lunak serta mikrokontroler ESP32. Model SVM dihubungkan dengan perangkat keras, yaitu motor servo sebagai penggerak pintu garasi, sensor *infrared* sebagai pendeteksi halangan, serta LED dan buzzer sebagai indikator status sistem. Komunikasi berlangsung melalui jaringan Wi-Fi sehingga hasil prediksi dapat diterjemahkan secara real-time menjadi aksi fisik pada perangkat keras.

Pengujian sistem dilakukan setelah implementasi selesai. Pengujian mencakup evaluasi performa identifikasi *speaker* menggunakan SVM, efektivitas pencocokan kata kunci dengan Regex, dan pengukuran waktu respons sistem dari masukan suara hingga perangkat keras bergerak.

Analisis hasil menjadi tahap terakhir dalam desain penelitian ini. Evaluasi dilakukan berdasarkan performa SVM dalam mengenali pengguna, akurasi Regex dalam membaca instruksi, tingkat error sistem, serta konsistensi respons perangkat keras. Analisis ini digunakan untuk menilai apakah sistem pintu garasi pintar telah berfungsi sesuai kebutuhan, mampu menjaga keamanan akses, dan tetap responsif terhadap variasi suara maupun kondisi lingkungan.

3.1.1. Pengumpulan Data

Pengolahan data dalam penelitian ini dilakukan untuk membangun dataset suara yang digunakan pada pelatihan dan pengujian sistem Garasi Pintar berbasis *voice recognition*. Data yang dikumpulkan terdiri atas dua kategori utama, yaitu identifikasi pengguna dan pencocokkan perintah suara. Untuk mendukung pendekatan *speaker-dependent*, dataset diklasifikasikan ke dalam dua folder, yakni *Me* yang berisi suara peneliti sendiri sebagai pengguna terdaftar, dan *Notme* yang berisi suara dari orang lain yang tidak terdaftar dalam sistem. Penjelasan lebih lengkap mengenai alur pengolahan suara dapat dilihat pada Gambar 3.2 di bawah ini.



Gambar 3.2. Alur Pengolahan Suara

Dataset terdiri atas tiga jenis ucapan, yaitu “Buka”, “Tutup”, dan *filler* berupa kata bebas yang berfungsi sebagai non-perintah. Kategori *Me* direkam sebanyak 90 sampel untuk kata “Buka”, 90 sampel untuk “Tutup”, dan 90 sampel *filler*, sehingga totalnya berjumlah 270 rekaman suara. Kategori *Notme* disusun dengan jumlah yang sama untuk menjaga keseimbangan distribusi kelas, yaitu 90 rekaman “Buka”, 90 rekaman “Tutup”, dan 90 rekaman *filler*, dengan total 270 data. Gabungan kedua kategori menghasilkan 540 berkas audio yang digunakan pada proses pelatihan dan pengujian.

Tabel 2.2. Variasi Kalimat yang Digunakan dalam Pelatihan

No	Jenis Ucapan	Contoh Kalimat	Keterangan
1	Buka	“buka”	Perintah inti
2	Buka	“tolong buka pintu garasinya”	Variasi Panjang
3	Buka	“buka pintunya sekarang”	Ungkapan langsung
4	Buka	“ayo buka pintunya”	Variasi informal
5	Buka	“buka pintunya cepat”	Aksen cepat
6	Tutup	“tutup pintunya dong”	Variasi Panjang
7	Tutup	“pintunya tolong tutup”	Struktur terbalik
8	Tutup	“cepat tutup pintunya”	Aksen cepat
9	Tutup	“tutup”	Perintah inti
10	Tutup	“tutup pintunya”	Variasi intonasi
11	*Filler*	“halo”	Ujaran non-perintah
12	*Filler*	Ucapan bebas dari responden	Random noise talk

Seluruh suara direkam menggunakan mikrofon eksternal pada jarak ±20 cm dalam ruangan dengan tingkat kebisingan rendah hingga sedang untuk menjaga rasio sinyal terhadap *noise*. Setiap rekaman disimpan dalam format *.wav* (PCM 16-bit, 16 kHz) dan dipisahkan ke dalam folder *Me* dan *Notme*. Jumlah sampel dirancang agar memenuhi kebutuhan representasi akustik yang stabil Kinnunen & Li (2010) menyatakan bahwa sistem *speaker verification* memerlukan 50–150 sampel per kelas agar distribusi fitur memadai. B. Davis & Mermelstein (1980) menambahkan bahwa 100–200 sampel per kata cukup untuk membangun model

MFCC yang konsisten. Jumlah 460 data yang dibagi seimbang antar kelas dinilai sesuai untuk melatih model SVM tanpa menimbulkan bias.

Analisis data dilakukan dengan mengekstraksi ciri suara menggunakan MFCC. Setiap rekaman menghasilkan vektor fitur yang kemudian digunakan sebagai input model. Data hasil ekstraksi dibagi ke dalam kategori *Me* dan *Notme* lalu dilatih menggunakan SVM untuk memisahkan pola akustik pada ruang fitur multidimensi. Evaluasi performa dilakukan menggunakan *error rate* yang dihitung dengan persamaan berikut:

$$Error\ Rate = \frac{Number\ of\ Request\ with\ Errors}{Total\ Number\ of\ Requests}\ x\ 100\% \qquad (3.1)$$

Jika suara berhasil diverifikasi sebagai *Me*, maka proses dilanjutkan dengan pencocokkan perintah menggunakan Regex untuk membedakan antara perintah "Buka" atau "Tutup".

3.1.2. Desain Sistem

Desain sistem Garasi Pintar berbasis pengenalan suara dibagi menjadi tiga tahapan utama, yaitu *input, process*, dan *output*. Ketiga tahapan ini menggambarkan alur kerja sistem mulai dari perintah suara yang diberikan pengguna, tahap pemrosesan untuk mengenali dan mencocokkan perintah, hingga menghasilkan keluaran berupa pergerakan pintu garasi. Selain itu, penelitian ini juga menyajikan rangkaian sistem untuk memperlihatkan keterhubungan antar komponen perangkat keras.

3.1.2.1. Tahapan *Input*

Tahapan *input* dimulai dari proses penangkapan suara oleh aplikasi *mobile* berbasis Flutter. Pengguna menekan tombol mikrofon untuk merekam perintah suara seperti “Tolong buka pintu garasi” atau “Tolong tutup pintu garasi”. Rekaman suara disimpan sementara sebagai file *.wav* yang kemudian dikirimkan ke *backend machine learning* menggunakan protokol HTTP. Penggunaan HTTP memberikan kemampuan pengiriman file audio dalam bentuk *multipart/form-data* melalui jalur komunikasi yang stabil, sehingga *backend* dapat menerima data suara secara utuh dan siap diproses. Protokol ini juga memungkinkan proses pengiriman berlangsung cepat karena karakteristiknya yang berbasis *request–response*, sehingga aplikasi dapat mengirimkan satu rekaman suara untuk segera diproses oleh server.

3.1.2.2. Tahapan *Process*

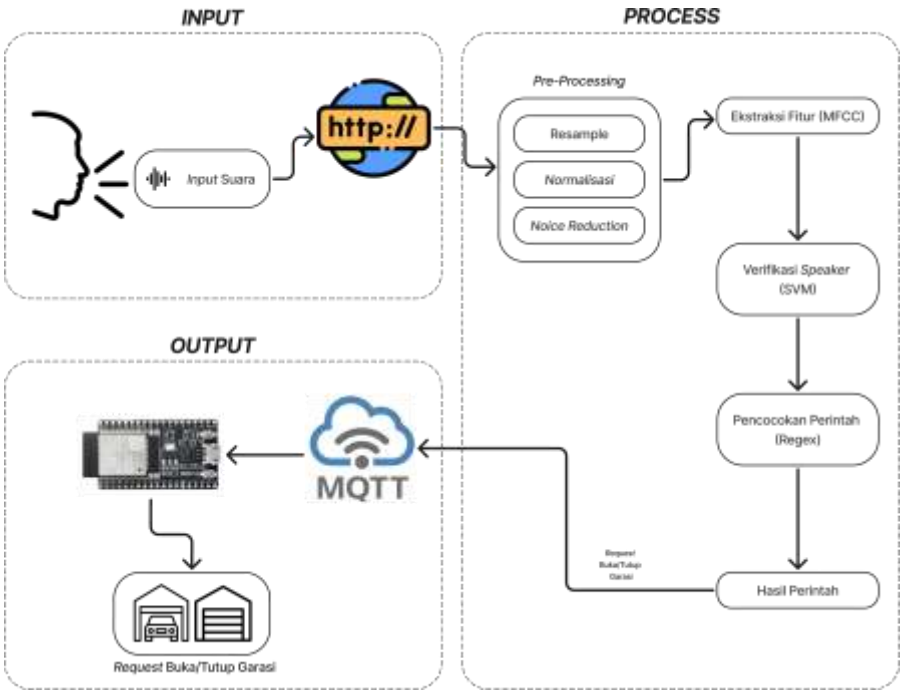
Tahapan proses merupakan inti dari sistem yang menangani seluruh pemrosesan data suara untuk menghasilkan keputusan akhir. *Backend* menerima file audio dari aplikasi melalui HTTP dan menjalankan rangkaian *pre-processing* yang meliputi resampling audio ke 16 kHz mono, normalisasi amplitudo, penghapusan noise menggunakan metode *spectral gating*, serta penerapan *pre-emphasis* untuk memperjelas komponen frekuensi tinggi. Audio yang telah dibersihkan diekstraksi menggunakan 13 koefisien MFCC untuk menghasilkan representasi numerik yang mencerminkan karakteristik suara pengguna. Fitur tersebut diverifikasi menggunakan algoritma SVM untuk memastikan bahwa suara berasal dari pengguna yang sah. Proses verifikasi dilanjutkan dengan pencocokan perintah menggunakan Regex untuk menentukan apakah instruksi yang diberikan

adalah “Buka” atau “Tutup”. Hasil dari seluruh proses ini dirumuskan sebagai keputusan yang kemudian dikirimkan ke perangkat ESP32 menggunakan protokol MQTT agar dapat dieksekusi oleh perangkat keras.

3.1.2.3. Tahapan *Output*

Tahapan *output* merupakan tahap akhir yang mengubah hasil keputusan dari *backend* menjadi eksekusi pada perangkat keras. Mikrokontroler ESP32 menerima pesan MQTT berisi instruksi pembukaan atau penutupan pintu garasi dan mengeksekusi perintah tersebut melalui komponen mekanik dan indikator visual. Servo bergerak membuka atau menutup pintu sesuai instruksi yang diterima. LED utama berfungsi sebagai indikator status alat yang selalu aktif selama sistem berjalan, sedangkan mode kedip pada LED digunakan untuk menandai bahwa mekanisme pintu sedang beroperasi, baik dalam proses membuka maupun menutup. Sistem keselamatan diimplementasikan menggunakan sensor *infrared* yang mendeteksi keberadaan objek pada jalur pintu. Buzzer hanya berbunyi ketika sensor mendeteksi halangan saat proses penutupan berlangsung. Pintu akan berhenti bergerak selama halangan terdeteksi untuk mencegah tabrakan. Gerakan menutup dilanjutkan kembali setelah jalur *infrared* dinyatakan bersih dan tidak ada objek yang menghalangi.

Untuk lebih jelas dapat dilihat pada desain sistem pada penelitian ini pada Gambar 3.3 di bawah ini:



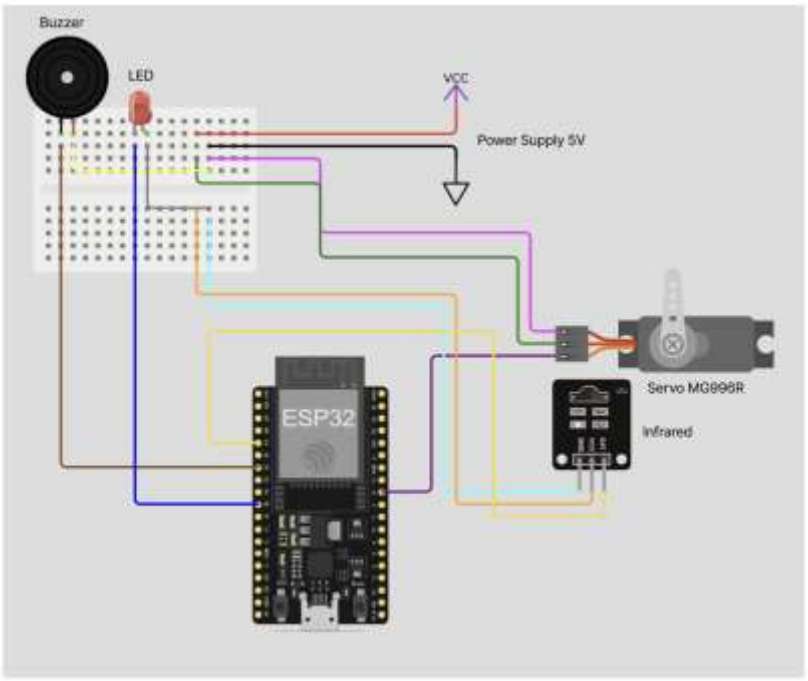
Gambar 3.3. Desain Sistem

3.1.2.4. Rangkaian Sistem

Rangkaian sistem Garasi Pintar dapat dilihat pada Gambar 3.4, yang menunjukkan alur integrasi komponen perangkat keras dan perangkat lunak. Proses perancangan dimulai dengan menentukan kebutuhan komponen utama seperti mikrofon, mikrokontroler ESP32, motor servo, buzzer, LED, serta sensor *infrared*. Setelah itu dilakukan penyusunan rangkaian, kalibrasi, dan pengujian untuk memastikan fungsi setiap komponen berjalan optimal. Jika terjadi ketidaksesuaian, dilakukan evaluasi dan penyesuaian sebelum melanjutkan ke tahap integrasi penuh.

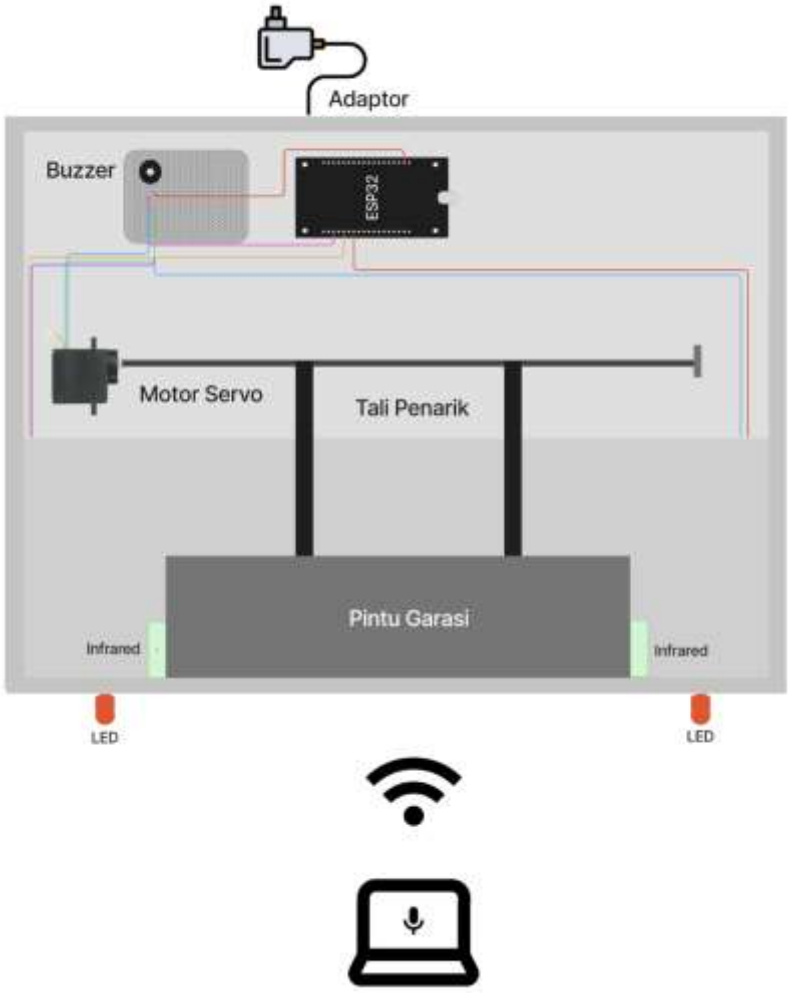
Gambar 3.4. Alur Rangkaian Sistem

Untuk implementasi fisik, digunakan *wiring schema* seperti pada Gambar 3.5. Sistem menggunakan adaptor 5V DC sebagai sumber daya utama, yang dialirkan ke *terminal block* untuk distribusi ke berbagai komponen. ESP32 berfungsi sebagai pusat kendali yang menerima perintah hasil pemrosesan suara dari laptop melalui jaringan Wi-Fi. Selanjutnya, ESP32 mengirimkan instruksi ke motor *servo* untuk membuka atau menutup pintu garasi. Pada saat yang sama, LED menyala sebagai indikator visual dan *buzzer* berbunyi sebagai indikator audio. Dengan demikian, setiap komponen dalam skema rangkaian memiliki fungsi spesifik, adaptor sebagai sumber daya, *terminal block* sebagai distribusi, ESP32 sebagai pengendali, motor *servo* sebagai aktuator, serta LED dan *buzzer* sebagai indikator status.



Gambar 3.5. *Wiring Schema*

Untuk memperlihatkan susunan fisik perangkat, implementasi rangkaian ditunjukkan pada Gambar 3.6. Komponen utama ditempatkan dengan perhitungan tertentu agar tidak mengganggu mekanisme pintu. ESP32, motor *servo*, *buzzer*, LED, serta sensor *infrared* diposisikan di bagian atas rangka utama agar mudah dihubungkan dengan batang pengait pintu. Sumber daya dari adaptor 5V DC dialirkan melalui *terminal block*, dengan jalur *ground* dibuat menyatu untuk menjaga kestabilan sinyal. Penataan kabel dilakukan secara rapi dengan pengikat agar aman dan estetis. Motor *servo* digunakan sebagai aktuator utama yang memutar batang pengait pintu garasi, sedangkan *buzzer* dan LED berfungsi sebagai indikator status sistem. Sensor *infrared* dipasang di sisi kiri pintu untuk mendeteksi adanya hambatan, sehingga sistem dapat menghentikan gerakan servo dan membuka kembali pintu jika terdeteksi objek.



Gambar 3.6. Implementasi Rangkaian

Secara keseluruhan, desain dan rangkaian sistem Garasi Pintar tidak hanya menekankan pada fungsi utama membuka dan menutup pintu berdasarkan perintah suara, tetapi juga memperhatikan aspek keamanan, kerapian, serta efisiensi penempatan komponen.

3.1.3. Penerapan Metode

Metode yang digunakan dalam penelitian ini bertujuan untuk merancang sistem Garasi Pintar berbasis perintah suara yang dapat digunakan secara mandiri, terutama untuk memudahkan akses dan meningkatkan keamanan pengguna.

Penerapan metode mencakup proses pengenalan suara, pencocokkan perintah, serta pengaktifan perangkat keras berdasarkan hasil pencocokkan tersebut. Seluruh proses dijalankan pada mikrokontroler ESP32 dan diprogram melalui Arduino IDE.

3.1.3.1. Pra-pemrosesan

Pra-pemrosesan diperlukan untuk menyamakan spesifikasi teknis seluruh data audio. Setiap berkas dikonversi ke format *.wav* dengan laju sampel 16 kHz dan mono channel sehingga memiliki frekuensi sampling yang seragam. Sinyal audio kemudian diubah menjadi deret diskrit amplitudo berdasarkan frekuensi sampling 16.000 Hz. Tahapan dilanjutkan dengan normalisasi amplitudo dengan membagi setiap sampel terhadap nilai absolut maksimum sinyal agar amplitudo berada pada rentang −1 hingga 1.

Durasi minimal input suara mengacu pada praktik umum dalam sistem pengenalan kata pendek (*short utterance*). Google Speech Commands Dataset sebagai standar internasional untuk *keyword spotting* menggunakan berkas audio berdurasi satu detik untuk setiap kata perintah (Warden, 2018), sehingga durasi satu detik dianggap memadai untuk menangkap ucapan tunggal seperti “buka” dan “tutup”. Penelitian ini menetapkan durasi rekaman maksimal lima detik agar seluruh variasi perintah, baik kata tunggal maupun kalimat pendek, tetap terekam tanpa risiko pemotongan (*truncation*) serta tetap berada dalam rentang optimal untuk pemrosesan *short utterance*. Adapun contoh perhitungan sampel sebagai berikut:

$$x_1 = \frac{-0.000000119}{0.6358286} = - 0.000000187 \quad (3.4)$$

Pra-pemrosesan dilanjutkan dengan reduksi *noise* menggunakan metode *spectral gating* berbasis ambang amplitudo. Nilai ambang diperoleh dari rata-rata amplitudo absolut (*noise_avg*). Sampel dengan amplitudo di bawah ambangdianggap sebagai *noise* dan dinolkan, sedangkan sampel di atas ambang dipertahankan. Persamaan reduksi *noise:*

$$x_{clean}[n] = \begin{cases} 0 & Jika\ |x_{norm}[n] < noice\_avg \\ x_{norm}[n], & Lainnya \end{cases} \quad (3.5)$$

Adapun contoh sampel sebagai berikut:

$$noice\_avg = 0.01933745921$$

Sampel pertama: $x_{norm}[0] = 1.84\ x\ 10^{-7}$

$$|x_{norm}[0]| = 0.000000184 < 0.019337$$

$$x_{clean}[0] = 0$$

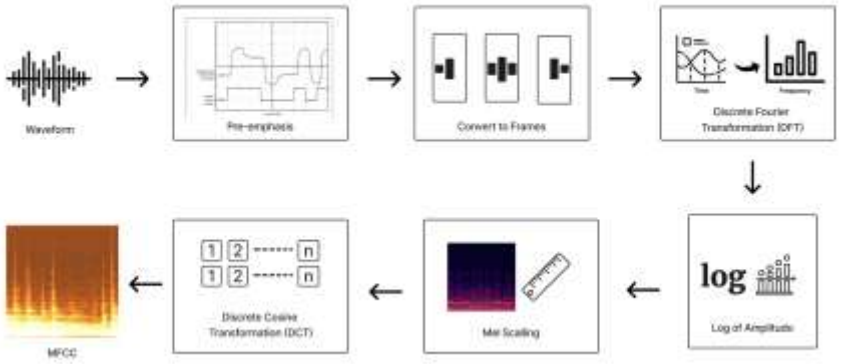
Sampel ke-7 : $x_{norm}[6] = -1.00$

$$|x_{norm}[6]| = 1.00 < 0.019337$$

$$x_{clean}[6] = 1.00$$

3.1.3.2. Ekstraksi Fitur dengan MFCC

Tahapan ekstraksi ciri menggunakan *Mel-Frequency Cepstral Coefficients* dilakukan setelah reduksi noise. MFCC digunakan karena mampu merepresentasikan karakteristik spektral suara sesuai skala Mel yang menyerupai pola persepsi telinga manusia.



Gambar 3.7. Proses Ekstraksi Fitur MFCC

Gambar 3.7 menjelaskan urutan proses ekstraksi. Tahapan dimulai dengan *waveform* sebagai representasi awal sinyal audio pada domain waktu. Sinyal kemudian diberi *pre-emphasis* untuk menonjolkan komponen frekuensi tinggi sehingga distribusi energi lebih seimbang (Heriyanto et al., 2018). Sinyal hasil filter dibagi menjadi *frame* karena sifat suara yang tidak stasioner. Setiap *frame* ditransformasikan menggunakan *Discrete Fourier Transform* (DFT) untuk memperoleh spektrum frekuensi.

Spektrum frekuensi selanjutnya dikonversi ke skala logaritmik untuk menyesuaikan persepsi telinga manusia terhadap intensitas suara. Proses berikutnya dilakukan Mel scaling menggunakan filter bank segitiga berbasis skala Mel untuk merepresentasikan karakteristik frekuensi sesuai dengan respons pendengaran manusia (Utama et al., 2025). Hasil proses tersebut kemudian diekstraksi menggunakan *Mel-Frequency Cepstral Coefficients* (MFCC). Pada penelitian ini digunakan 13 koefisien MFCC pada setiap frame, karena koefisien tersebut merupakan koefisien dominan yang mampu merepresentasikan karakteristik spektral utama sinyal suara secara efektif. Pemilihan 13 koefisien MFCC juga

banyak digunakan dalam penelitian pengenalan identitas penutur karena memberikan keseimbangan antara representasi fitur yang informatif dan kompleksitas komputasi yang efisien(Sasongko et al., 2023).

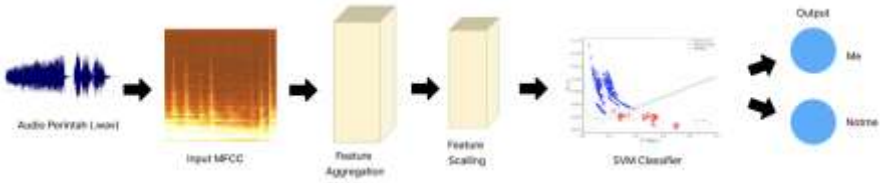
Nilai koefisien MFCC dari setiap *frame* dihitung rata-ratanya menggunakan metode *mean pooing* untuk membentuk vector fitur berukuran tetap yang kemudian digunakan sebagai masukan ke dalam model SVM. Hasil ekstraksi koefisien ditunjukkan pada Tabel 3.1 berikut:

Tabel 3.1. Hasil Ekstraksi Suara

Filename	Speaker	Command	MFCC_1	MFCC_...	MFCC_12	MFCC_13
Buka 2.wav	Notme	Buka	-622.924	…	-1.92221	-3.31385
Buka 2.wav	Notme	Buka	-598.953	…	0.128697	-10.9951
Tutup 4.wav	Me	Tutup	-666.881	…	3.295543	5.607557
Tutup 4.wav	Me	Tutup	-538.667	…	8.234294	5.244555
Tutup 4.wav	Me	Tutup	-479.106	…	6.553525	4.489096

3.1.3.3. Pengenalan *Speaker* dengan *Support Vector Machine*

Tahap pengenalan *speaker* bertujuan membedakan apakah suara berasal dari pengguna sah (*Me*) atau bukan pengguna (Notme). Rekaman suara dalam format *.wav* dikonversi menjadi representasi digital yang siap diekstraksi fiturnya. Setiap rekaman menghasilkan 13 koefisien MFCC per *frame* yang direpresentasikan sebagai karakteristik frekuensi suara. Seluruh koefisien MFCC digabungkan melalui proses feature aggregation untuk membentuk vektor fitur representatif per rekaman.



Gambar 3.8. Model Arsitektur SVM

Gambar 3.8 menggambarkan arsitektur SVM. Vektor fitur dinormalisasi menggunakan *StandardScaler* agar memiliki distribusi dengan rata-rata nol dan standar deviasi satu. Data kemudian dibagi menjadi data latih dan data uji sebelum pemodelan dilakukan. Algoritma yang digunakan adalah SVM dengan dua jenis kernel, yaitu Linear dan *Radial Basis Function* (RBF). Kernel Linear digunakan untuk melihat kemampuan pemisahan data dalam ruang fitur yang bersifat mendekati linear (Handoko & Suyanto, 2019), sedangkan kernel RBF digunakan untuk mengevaluasi performa ketika pola pemisahan lebih kompleks dan memerlukan pemetaan ke dimensi yang lebih tinggi. Kedua kernel ini diuji untuk memperoleh model dengan performa terbaik dalam membedakan suara *Me* dan *Notme*.

Secara matematis, fungsi keputusan SVM dinyatakan sebagai:

$$f(x) = w^T x + b, \quad y = sign(f(x)) \tag{3.6}$$

Model bekerja dengan menentukan hyperplane terbaik untuk memisahkan suara *Me* dan *Notme*. Fungsi kernel RBF digunakan untuk memetakan data ke ruang berdimensi lebih tinggi sehingga pola antar kelas lebih mudah dipisahkan. Dengan pendekatan ini, proses keputusan SVM dilakukan berdasarkan fungsi:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b \tag{3.7}$$

Jika hasil perhitungan $f(x)$ bernilai positif maka suara dikenal sebagai *Me*, sedangkan jika negatif maka dikenal sebagai *Notme*. Nilai $f(x)$ juga dapat dikonversi menjadi probabilitas untuk menunjukkan seberapa yakin model terhadap hasil prediksinya. Dengan pendekatan ini, sistem dapat mengidentifikasi speaker secara *real-time*, memanfaatkan karakteristik spektrum suara masing-masing pengguna untuk membedakan suara pengguna yang sah dari suara orang lain.

3.1.3.4. Pencocokan Perintah dengan Regex

Pengenalan perintah dilakukan setelah suara diverifikasi sebagai pengguna sah. Proses dimulai dari konversi sinyal suara menjadi teks melalui mekanisme *Speech-to-Text* (STT). Model akustik, model bahasa, dan proses decoding bekerja bersama untuk menghasilkan transkripsi. Teks hasil STT dinormalisasi terlebih dahulu untuk menyamakan format.

Regex digunakan untuk menemukan pola kata atau frasa yang berkaitan dengan instruksi yang dimaksud, misalnya pola *buka* dan *tutup*. Kombinasi STT dan regex menjadikan proses pemetaan perintah lebih transparan dan mudah ditelusuri apabila terjadi kesalahan.

3.1.3.5. Eksekusi Perintah

Perintah yang tervalidasi akan diterjemahkan ESP32 menjadi aksi perangkat keras. Instruksi “Buka” memicu servo untuk membuka pintu, sedangkan instruksi “Tutup” memicu servo untuk menutupnya. ESP32 mengendalikan servo, LED, dan buzzer sehingga pengguna mendapatkan umpan balik visual dan audio terhadap status sistem.

3.1.4. Uji Coba Sistem

Pengujian pada penelitian ini dilakukan untuk mengevaluasi kinerja sistem Garasi Pintar berbasis *voice recognition* yang dikembangkan menggunakan algoritma SVM untuk pengenalan pengguna dan Regex untuk pencocokkan perintah suara. Skenario pengujian dirancang agar dapat menunjukkan sejauh mana sistem mampu mengenali pola suara pengguna dan menjalankan perintah dengan benar sesuai konteks.

3.1.4.1. Pengujian Pengenalan *Speaker*

Tahapan ini bertujuan untuk mengukur kemampuan sistem dalam membedakan antara suara pengguna sah (kelas *Me*) dan bukan pengguna (*Notme*). Model SVM dilatih menggunakan dataset suara yang telah melalui proses *pre-processing* dan ekstraksi fitur MFCC. Dataset dibagi menjadi dua bagian, yaitu data latih (*training* set) dan data uji (*testing* set), dengan beberapa variasi rasio pembagian untuk melihat kestabilan performa model. Parameter utama yang diuji adalah nilai *C* (*regularization parameter*) dan gamma menggunakan kernel RBF. Pengujian dilakukan dengan metode *grid search* dan *cross-validation* untuk menentukan konfigurasi terbaik.

Tabel 3.2. Tabel Pengujian *Speaker* dengan SVM

Skenario	Rasio Data Latih	Rasio Data Uji	Nilai C	Nilai Gamma (γ)	Tujuan
1	90%	10%	0.1, 1, 10	0.1, 0.5, 1	Melihat performa model pada rasio umum yang sering digunakan.
2	80%	20%	0.1, 1, 10	0.1, 0.5, 1	Menguji ketahanan model dengan data uji lebih besar.

3	70%	30%	0.1, 1, 10	0.1, 0.5, 1	Mengevaluasi stabilitas model dengan data latih yang lebih sedikit.

Hasil prediksi model pada setiap skenario nantinya akan dibandingkan dengan label aktual dan dievaluasi menggunakan confusion matrix untuk menghitung metrik *accuracy*, *precision*, *recall*, dan *F1-score*.

3.1.4.2. Pengujian Pencocokkan Perintah Suara

Tahap ini bertujuan untuk menguji kemampuan sistem dalam mengenali dan mengeksekusi perintah pengguna melalui mekanisme Regex. Setelah proses identifikasi *speaker* berhasil, sistem akan melakukan pencocokan teks hasil *speech-to-text* dengan pola Regex yang telah dirancang untuk mengenali kata kunci seperti "buka" dan "tutup" pada berbagai bentuk kalimat.

Pengujian dilakukan dengan memberikan variasi ucapan yang memiliki makna sama tetapi dengan struktur kalimat yang berbeda, guna menguji fleksibilitas pola Regex dalam menangani variasi bahasa alami pengguna.

Tabel 3.3. Tabel Pengujian Pencocokkan Perintah dengan *Regular Expression*

Skenario	Jenis Ucapan	Pola Regex	Tujuan
1	Ucapan sederhana ("buka garasi")	`buka	Open'
2	Ucapan kompleks ("tolong buka pintu garasi")	buka.*garasi	Menguji efektivitas Regex terhadap kalimat panjang.
3	Ucapan tidak relevan ("bagaimana cuacanya")	—	Menguji mekanisme *fallback* terhadap perintah yang tidak dikenali.

3.1.5. Hasil Analisis

Pengujian ini dilakukan untuk memastikan seluruh komponen sistem, mulai dari proses perekaman suara, identifikasi pengguna dengan algoritma SVM kernel linear, pencocokan perintah menggunakan Regex, hingga eksekusi servo pada garasi, dapat bekerja secara terpadu dan menghasilkan respon yang sesuai.

Uji coba sistem dilakukan secara semi-riil dengan melibatkan 7 responden sebagai pengguna. Setiap responden memberikan 20 kali input suara yang mencakup variasi perintah dan kondisi pengujian, sehingga total terdapat 140 sampel pengujian. Tujuan pengujian ini adalah untuk menilai sejauh mana sistem mampu mengenali perintah secara konsisten dari berbagai karakteristik suara dan intonasi.

Kondisi pengujian dibagi ke dalam beberapa skenario berikut:

Tabel 3.4. Skenario Pengujian Integrasi Sistem

Skenario	Kondisi Lingkungan	Jumlah Responden	Jumlah Uji per Responden	Total Sampel	Tujuan
1	Suara jelas tanpa *noise*	7	20	140	Menguji sistem pada kondisi ideal.
2	Suara dengan intonasi berbeda	7	20	140	Menguji toleransi sistem terhadap variasi intonasi.
3	Suara dengan jarak ±0.5 m dari mikrofon	7	20	140	Menguji pengaruh jarak terhadap pengenalan suara.
4	Suara dengan latar belakang noise rendah	7	20	140	Menguji stabilitas

					sistem terhadap gangguan lingkungan ringan.
5	Perintah kompleks atau tidak langsung (“tolong buka pintu garasi”)	7	20	140	Menguji fleksibilitas Regex dalam mengenali variasi kalimat perintah.

Total seluruh pengujian adalah 140 kali percobaan, yang terdiri atas lima kondisi berbeda dengan masing-masing melibatkan 7 responden yang masing-masing memberikan 20 *input* suara dalam lima kondisi berbeda. Setiap skenario dirancang untuk merepresentasikan variasi lingkungan dan pola bicara yang mungkin terjadi pada penggunaan nyata, guna memastikan sistem mampu bekerja secara andal di berbagai situasi akustik.

BAB IV
HASIL DAN PEMBAHASAN

4.1. Pengumpulan Data

Data penelitian ini diperoleh melalui proses perekaman suara dari 18 partisipan yang setiap partisipan diminta mengucapkan tiga jenis kalimat, yaitu perintah "Buka", "Tutup", dan kalimat bebas (*filler*). Seluruh perekaman dilakukan menggunakan mikrofon eksternal pada jarak sekitar ±20 cm dari mulut di ruangan dengan tingkat kebisingan rendah hingga sedang. Rekaman suara disimpan dalam format *.m4a* sebagai data mentah (*raw data*), dengan jumlah dan struktur yang berbeda antara kategori *Me* (suara pengguna sah) dan *Notme* (suara penutur lain). Dataset *Me* direkam oleh satu orang pengguna utama, sedangkan dataset *Notme* berasal dari 18 orang partisipan lain.

Setiap partisipan *Notme* diminta mengucapkan perintah "Buka" dan "Tutup" masing-masing sebanyak 5 kali, serta beberapa kalimat bebas (*filler*) sebanyak ±4–5 kali. Sementara itu, untuk pengguna sah (Me), perekaman dilakukan dengan jumlah yang lebih banyak, yaitu 90 kali untuk perintah "Buka", 90 kali untuk perintah "Tutup", dan 90 kalimat *filler* yang berisi ujaran relevan seperti "Tolong buka pintunya".

Hasil keseluruhan perekaman mentah ditunjukkan pada Tabel 4.1 berikut.

Tabel 4.1. Distribusi Data Penelitian

Kategori Speaker	Jenis Perintah	Jumlah Data
Me	Buka	90
Me	Tutup	90
Me	*Filler*	90
Notme	Buka	90
Notme	Tutup	90
Notme	*Filler*	90
Total		**540**

Seluruh data mentah ini kemudian diproses melalui tahap *pre-processing* yang mencakup konversi format menjadi *.wav* mono 16 kHz, pembersihan sinyal melalui penyaringan dan *noise reduction*, normalisasi amplitudo, serta penyeragaman durasi menjadi 5 detik sebelum digunakan dalam proses ekstraksi fitur MFCC dan pelatihan model.

4.2. Implementasi Metode

Pada penelitian ini, proses identifikasi dilakukan melalui beberapa tahapan utama, yaitu *pre-processing* audio, ekstraksi fitur menggunakan MFCC, dan pengenalan suara menggunakan model berbasis SVM. Setiap tahap memiliki perannya masing-masing dalam memastikan bahwa sinyal suara yang dianalisis memiliki kualitas yang baik, fitur yang representatif, serta model yang mampu membedakan antara suara *Me* dan *Notme* secara akurat.

4.2.1. *Pre-Processing* Audio

Pre-processing dilakukan untuk menyeragamkan dan membersihkan sinyal sebelum ekstraksi fitur. Pada tahap ini, rekaman diselaraskan ke format yang konsisten (16 kHz, mono), dikondisikan secara spektral agar fokus pada pita wicara, bagian hening serta derau latar diturunkan, lalu level dan durasi diseragamkan.

Hasilnya adalah sinyal siap pakai yang stabil, homogen, dan representative sehingga proses ekstraksi MFCC dan tahap berikutnya dapat berjalan lebih akurat dan terukur.

4.2.1.1. Standarisasi

Seluruh rekaman dikonversi menjadi audio mono dengan laju sampel 16 kHz. Penyelarasan format ini memastikan setiap sinyal memiliki resolusi dan struktur yang sama sehingga perbedaan perangkat perekam tidak memengaruhi hasil analisis. Sinyal kemudian difokuskan pada rentang frekuensi wicara dengan menerapkan penyaringan pita (*band-pass*). Frekuensi yang terlalu rendah maupun terlalu tinggi dihilangkan karena umumnya berisi derau atau komponen non-wicara. Setelah itu dilakukan pengurangan derau untuk menekan *noise* latar yang bersifat statis sehingga bagian ujaran menjadi lebih jelas.

4.2.1.2. Normalisasi

Normalisasi dilakukan untuk menyamakan tingkat energi antarrekaman sehingga variasi *loudness* tidak memengaruhi proses ekstraksi fitur. Energi sinyal dihitung menggunakan *Root Mean Square* (RMS), kemudian amplitudo disesuaikan menuju nilai RMS target (γ_{tgt}). Dengan cara ini, seluruh rekaman memiliki level amplitudo yang seragam dan fitur yang dihasilkan menjadi lebih stabil. Gain normalisasi dihitung menggunakan persamaan:

$$g = \frac{\gamma tgt}{\sqrt{\frac{1}{N}\sum_{n=1}^{N} x[n]^2}} \tag{4.1}$$

Hasil normalisasi memastikan bahwa perbandingan antarrekaman tidak bias oleh perbedaan volume.

4.2.1.3. Penyeragaman Durasi

Setelah dinormalisasi, setiap sinyal diseragamkan panjangnya menjadi 5 detik. Rekaman yang lebih panjang dipangkas pada bagian akhir, sedangkan rekaman yang lebih pendek ditambahkan keheningan (*zero-padding*) hingga mencapai durasi yang sama. Penyeragaman durasi ini memastikan konsistensi jumlah *frame* pada tahap ekstraksi fitur dan memudahkan proses pelatihan model.

Untuk memastikan bahwa hasil *pre-processing* telah menghasilkan sinyal yang stabil dan representatif, dilakukan pengukuran terhadap karakteristik dasar seluruh 540 berkas audio yang telah dinormalisasi dan diseragamkan durasinya. Nilai rata-rata karakteristik sinyal ditunjukkan pada Tabel 4.2 berikut.

Tabel 4.2. Karakteristik Sinyal Hasil *Pre-Processing*

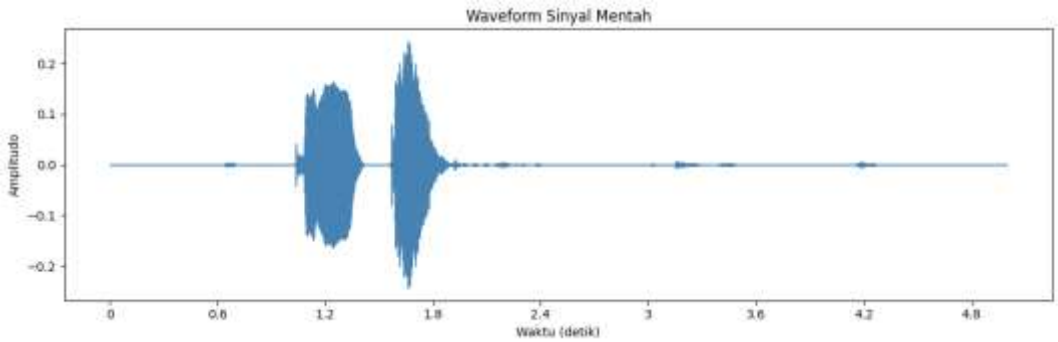
Parameter	Nilai Rata-rata
Amplitudo Maksimum	0.750217
RMS (*Root Mean Square*)	0.026686
Energi rata-rata (dB)	-31.74 dB
Frekuensi Dominan	1379.94 Hz

Berdasarkan hasil tersebut, dapat disimpulkan bahwa sinyal yang telah melalui tahap *pre-processing* memiliki tingkat energi dan frekuensi dominan yang berada dalam rentang karakteristik wicara manusia, yaitu sekitar 300–3400 Hz. Nilai amplitudo dan RMS yang relatif seragam menunjukkan bahwa proses normalisasi dan penyeragaman durasi berjalan efektif dalam menghasilkan dataset yang stabil dan layak digunakan pada tahap ekstraksi fitur MFCC dan proses identifikasi berikutnya.

4.2.2. Ekstraksi MFCC

Sebelum tahap pengenalan suara, setiap sinyal suara dikonversi menjadi representasi fitur numerik menggunakan *Mel-Frequency Cepstral Coefficients* (MFCC). MFCC menangkap bentuk selubung spektrum (*spectral envelope*) yang berkorelasi kuat dengan karakteristik artikulatoris dan persepsi manusia.

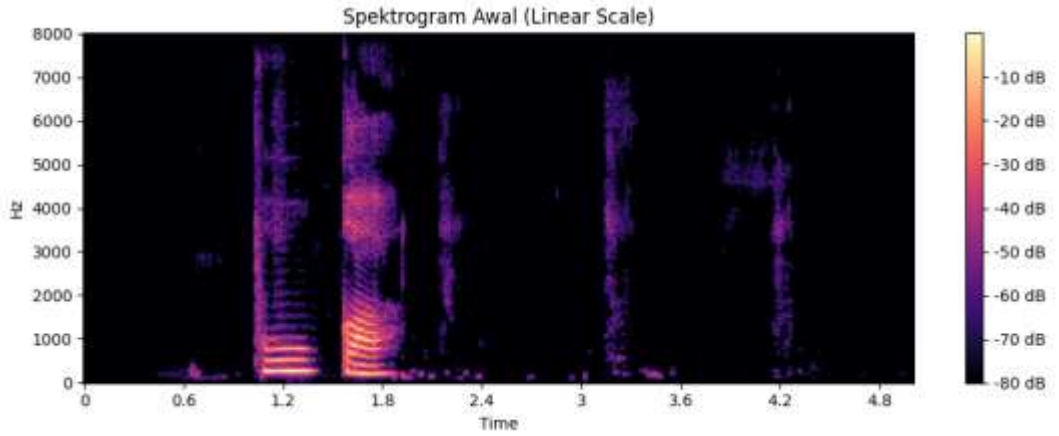
Proses ekstraksi MFCC diawali dengan pengamatan terhadap sinyal suara dalam ranah waktu dan frekuensi untuk memahami karakteristik dasar energi ujaran.



Gambar 4.1. *Waveform* Sinyal Suara Mentah

Pada Gambar 4.1 ditampilkan *waveform* sinyal suara mentah dengan durasi sekitar 5 detik. Sumbu horizontal (sumbu-x) merepresentasikan waktu dalam satuan detik, sedangkan sumbu vertikal (sumbu-y) menunjukkan amplitudo dari sinyal audio. Terlihat bahwa pada awal rekaman (0–1 detik) amplitudo berada sangat dekat dengan nol, yang menandakan adanya bagian hening sebelum penutur mulai mengucapkan perintah. Aktivitas suara mulai muncul secara jelas pada sekitar detik ke-1,2 hingga 1,9, ditandai oleh peningkatan amplitudo yang signifikan. Bagian ini merupakan inti dari ucapan penutur dan berisi energi wicara yang dominan. Setelah bagian tersebut, amplitudo kembali menurun mendekati nol, menandakan berakhirnya ujaran dan kembali ke kondisi hening. Pola ini menunjukkan secara

jelas kapan ujaran dimulai dan diakhiri, serta distribusi energi pada sepanjang rekaman.



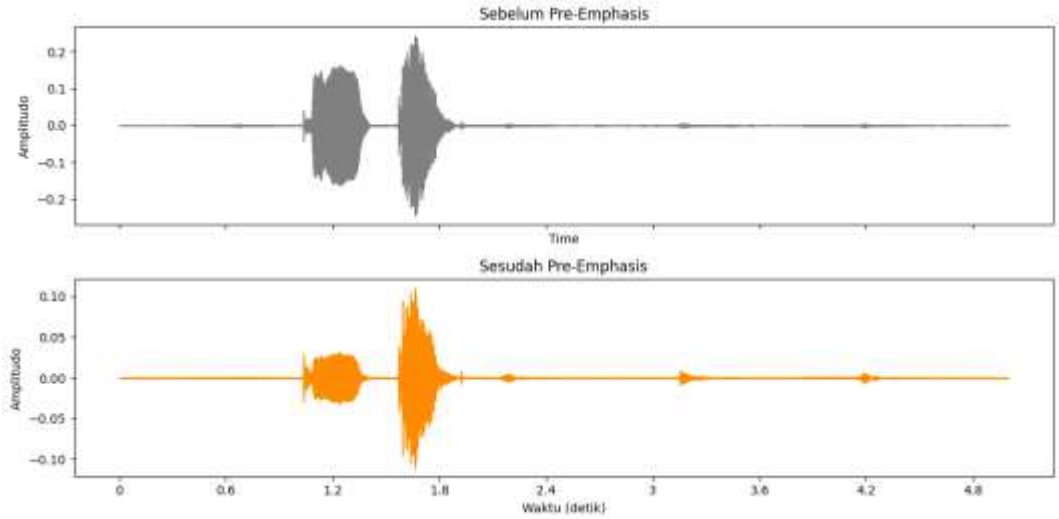
Gambar 4.2. Spektogram Suara Mentah

Pada Gambar 4.2 ditampilkan spektrogram dari sinyal suara mentah dalam skala linear. Sumbu horizontal (waktu) menunjukkan perkembangan sinyal sepanjang durasi 5 detik, sedangkan sumbu vertikal merepresentasikan frekuensi dalam satuan Hertz (Hz). Warna pada spektrogram menunjukkan intensitas energi frekuensi, di mana warna lebih cerah menandakan energi yang lebih tinggi dan warna lebih gelap menunjukkan energi rendah atau bagian yang mendekati keheningan. Terlihat bahwa aktivitas suara utama terjadi pada rentang waktu sekitar 1.1 hingga 1.9 detik, ditunjukkan oleh area berwarna lebih terang yang memanjang secara vertikal. Pola energi yang muncul pada pita frekuensi sekitar 300–4000 Hz yang merupakan ciri khas komponen wicara manusia.

4.2.2.1. *Pre-Emphasis*

Pre-emphasis adalah penyaringan linear orde-satu pada ranah waktu untuk menekankan komponen frekuensi tinggi dan menyelaraskan level frekuensi

rendah–tinggi. Digunakan untuk mengurangi derau, memperbaiki SNR pita tinggi, dan menyiapkan sinyal ke tahap berikutnya.



Gambar 4.3. Perbandingan Sinyal Sebelum dan Sesudah Proses *Pre-Emphasis*

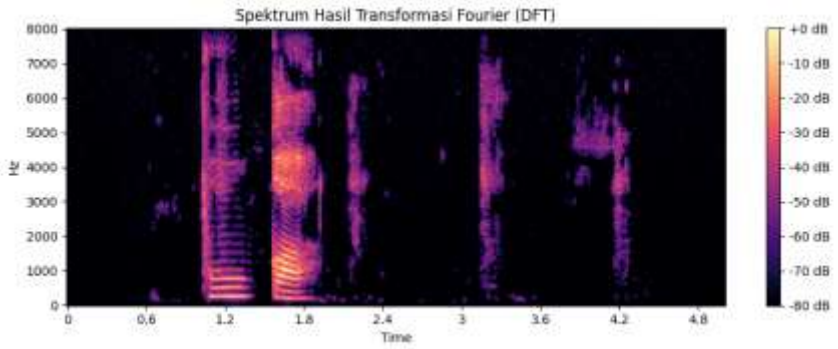
Pada Gambar 4.3 ditampilkan perbandingan bentuk sinyal suara sebelum dan sesudah diterapkan *pre-emphasis*. Pada bagian sebelum *pre-emphasis*, gelombang tampak lebih "halus" dan energi lebih terkonsentrasi pada komponen amplitudo besar yang berasal dari frekuensi rendah. Setelah proses *pre-emphasis* diterapkan, terlihat bahwa kontur gelombang menjadi lebih tajam dan lebih padat, terutama pada bagian transisi cepat dari sinyal. Hal ini menunjukkan peningkatan energi pada komponen frekuensi tinggi.

4.2.2.2. *Convert to Frames*

Convert to Frames merupakan pemotongan sinyal hasil pre-emphasis menjadi bingkai berdurasi pendek (±20–40 ms) dengan *overlap*, sehingga tiap bingkai dapat dianggap *quasi-stasioner*. Digunakan untuk memungkinkan analisis spektral yang representatif per interval waktu.

4.2.2.3. *Discrete Fourier Transformation*

Discrete Fourier Transformation merupakan transformasi dari domain waktu ke domain frekuensi untuk memperoleh spektrum (magnitudo/daya) tiap *frame*. Digunakan untuk memetakan kandungan sinusoidal sebagai dasar pemfilteran pada skala Mel.



Gambar 4.4. Spektogram Hasil Transformasi *Fourier*

4.2.2.4. *Log of Amplitude*

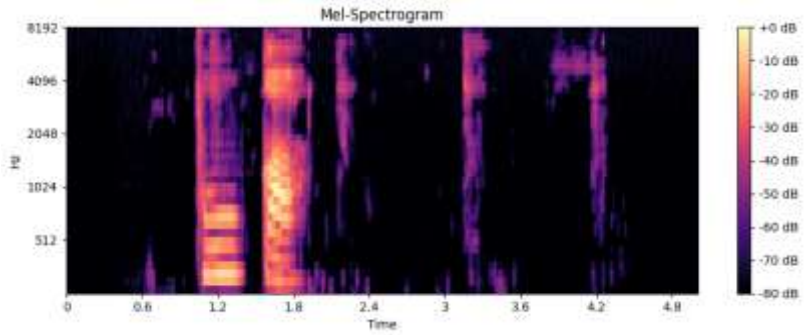
Log of Amplitude merupakan pemetaan energi spektral (atau energi pita-Mel) ke skala log/dB. Digunakan untuk menyelaraskan rentang dinamis dengan persepsi manusia dan menormalkan kontribusi puncak energi sebelum DCT.

4.2.2.5. *Mel Scaling*

Mel Scaling merupakan proses pemetaan spektrum linier hasil transformasi Fourier ke dalam skala perseptual Mel yang lebih menyerupai cara telinga manusia mendengar suara. Dalam skala Mel, resolusi frekuensi dibuat lebih rapat pada frekuensi rendah dan lebih renggang pada frekuensi tinggi, karena manusia lebih sensitif terhadap perubahan nada di pita rendah (sekitar 300–1000 Hz) dibandingkan pita tinggi.

Proses ini dilakukan menggunakan sekumpulan filter segitiga tumpang tindih (*Mel filter bank*), di mana masing-masing filter menghitung energi rata-rata

pada rentang frekuensi tertentu. Energi dari hasil *Discrete Fourier Transformation* (DFT) yang semula masih linier kemudian diproyeksikan ke skala Mel untuk menekankan komponen-komponen yang lebih relevan secara fisiologis terhadap persepsi manusia.



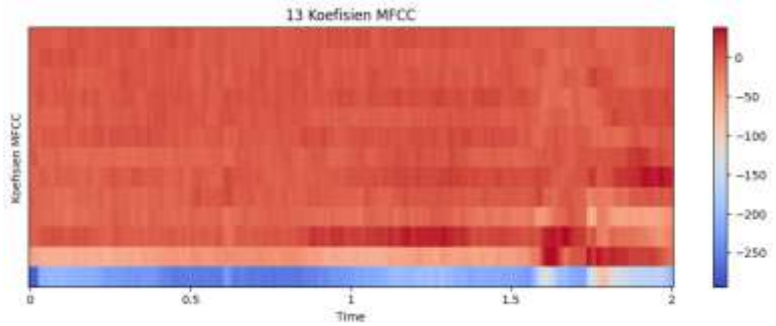
Gambar 4.5. Mel-Spectogram Hasil Pemetaan Frekuensi ke Skala Mel

Gambar 4.5 menunjukkan *Mel-Spectrogram* dari sinyal suara setelah spektrum linier diproyeksikan ke skala Mel. Warna yang lebih terang menggambarkan energi yang lebih tinggi pada frekuensi tertentu. Pola energi utama terlihat pada rentang sekitar 500–4000 Hz, yaitu daerah yang paling relevan untuk komponen wicara manusia. Pada rentang waktu 1,2–1,9 detik, tampak pita energi yang jelas sebagai bagian inti dari ujaran. Dibandingkan spektrogram linier, Mel-*Spectrogram* menampilkan detail yang lebih padat pada frekuensi rendah dan lebih ringkas pada frekuensi tinggi, sehingga pola formant dan karakteristik artikulasi lebih mudah diamati dan lebih representatif untuk proses ekstraksi MFCC.

4.2.2.6. *Discrete Cosine Transformation*

Discrete Cosine Transformation merupakan transformasi yang berfungsi untuk mendekorrelasi log-energi hasil *Mel-Spectrogram* dan memadatkannya menjadi sejumlah kecil koefisien yang merepresentasikan bentuk selubung spektral

dari sinyal suara. Dengan demikian, DCT mengubah informasi energi dalam domain frekuensi menjadi representasi *cepstral* yang lebih kompak, stabil, dan mudah dianalisis untuk proses identifikasi *speaker*. Dalam penelitian ini digunakan 13 koefisien MFCC sebagai fitur utama, di mana setiap koefisien menggambarkan variasi spektrum pada tingkat tertentu. Koefisien-ke-1 mewakili energi keseluruhan, sedangkan koefisien selanjutnya menangkap perbedaan karakteristik suara antar frekuensi yang bersifat unik.



Gambar 4.6. Representasi 13 Koefisien MFFC

Gambar 4.6 memperlihatkan representasi 13 koefisien MFCC yang dihasilkan dari sinyal berdurasi 5 detik. Sumbu horizontal menunjukkan waktu, sedangkan sumbu vertikal menunjukkan indeks koefisien MFCC. Warna pada gambar menggambarkan nilai masing-masing koefisien, di mana warna merah menunjukkan nilai yang lebih tinggi dan warna biru menunjukkan nilai yang lebih rendah. Pola perubahan warna sepanjang waktu mencerminkan dinamika artikulasi penutur selama mengucapkan perintah, terutama pada rentang waktu sekitar 1.2 hingga 1.9 detik, ketika energi wicara muncul. Koefisien-koefisien inilah yang merangkum bentuk selubung spektral suara dan menjadi dasar bagi sistem untuk membedakan karakteristik suara antara kelas *Me* dan *Notme* pada tahap pengenalan *speaker*.

Tabel 4.3. Hasil Ekstraksi MFCC

Index_sample	MFCC_1	MFCC_2	MFCC_3	…	MFCC_13	Kelas
0	0.000000114	0.000000000	-0.000000009	…	0.000000000	Me
1	-0.000000418	0.000000057	-0.000000066	…	0.000000019	Me
2	0.000000304	0.000000076	-0.000000026	…	0.000000000	Me

Tabel 4.3 menampilkan contoh hasil ekstraksi fitur MFCC dalam bentuk nilai numerik. Setiap kolom MFCC_1 hingga MFCC_13 merepresentasikan satu dimensi fitur *cepstral* yang dihasilkan dari proses DCT terhadap energi *Mel-Spectrogram.* Nilai pada masing-masing koefisien menunjukkan kontribusi relatif komponen spektral pada tingkat orde tertentu, di mana perbedaan nilai antar koefisien dan antar sampel membentuk pola karakteristik suara penutur. Koefisien MFCC_1 hingga MFCC_3 merepresentasikan komponen global selubung spektral, sehingga nilai-nilainya cenderung lebih stabil dan mencerminkan bentuk spektrum secara umum. Koefisien MFCC_4 hingga MFCC_7 menangkap variasi resonansi spektral yang berkaitan dengan karakteristik saluran vokal, yang menyebabkan nilai-nilainya bervariasi antar penutur. Sementara itu, koefisien MFCC_8 hingga MFCC_13 merepresentasikan detail spektral yang lebih halus, dengan nilai yang relatif kecil namun tetap berkontribusi dalam membedakan pola suara pada proses klasifikasi.

4.2.3. *Pre-Processing* Data

Tahap *pre-processing* merupakan langkah penting untuk menyiapkan data sebelum proses pelatihan model pengenalan suara. Tahapan ini memastikan data dalam kondisi bersih, seimbang, serta berada dalam skala yang seragam sehingga dapat diolah secara optimal oleh algoritma SVM.

Proses *pre-processing* ini dilakukan setelah tahap ekstraksi ciri (*feature extraction*) menggunakan MFCC, sehingga setiap berkas audio telah diubah menjadi representasi numerik fitur yang siap diproses oleh model pembelajaran mesin.

4.2.3.1. *Cleaning*

Tahap *cleaning* bertujuan memastikan hanya data yang valid dan relevan yang digunakan dalam proses *training*. Data hasil ekstraksi MFCC sering kali mengandung nilai kosong (*missing values*) atau label yang tidak sesuai akibat kesalahan anotasi atau kegagalan ekstraksi suara. Proses *cleaning* dilakukan dengan cara:

1. Menghapus baris data yang memiliki nilai kosong (NaN) pada kolom label maupun fitur.
2. Menyaring hanya label yang valid, yaitu *Me* dan *Notme*, dan menghapus label lain seperti "*unknown*" atau hasil pengenalan yang gagal.
3. Semua nilai pada kolom *label_speaker* diseragamkan ke huruf kecil untuk menghindari inkonsistensi format, misalnya perbedaan penulisan "Me", "me", atau "ME".

Adapun hasil dari sebelum dan sesudah dilakukan *cleaning* adalah sebagai berikut:

Tabel 4.3. Proses *Cleaning* pada Data

	Sebelum *Cleaning*		**Setelah *Cleaning***	
	Jumlah Data	**Data Hilang (NaN)**	**Jumlah Data**	**Data Hilang (NaN)**
Label Me	270	0	270	0
Label Notme	270		270	
Total Data	540		540	

Berdasarkan hasil pada Tabel 4.3, jumlah data awal sebanyak 540 sampel, terdiri dari 270 data kelas *Me* dan 270 data kelas *Notme*. Tidak ditemukan data yang hilang (*missing value*), sehingga jumlah data sebelum dan sesudah pembersihan tetap sama.

4.2.3.2. *Normalisai*

Normalisasi fitur adalah proses penskalaan agar skala antar dimensi fitur setara. Tujuan normalisasi adalah mencegah fitur beramplitudo besar. Penelitian ini menggunakan *StandardScaler* **(***Z-Score Normalization*) karena SVM peka terhadap skala dan *z-score* menjaga informasi jarak relatif dengan memusatkan data pada nol dan menyetarakan standar deviasi menjadi satu.

$$z_j = \frac{x_j - \mu_j}{\sigma_j} \tag{4.2}$$

Keterangan:

z_j : fitur yang telah dinormalisai

x_j : nilai fitur pada dimensi ke-j

μ_j : rata-rata fitur ke-j

σ_j : simpangan baku fitur ke-j

Contoh perhitungan:

$$z_{mfcc13} = \frac{-0.000000019 - 0.0000000322127}{0.0000000322127} = -0.61 \tag{4.3}$$

Proses normalisasi dapat dilihat melalui *pseudocode berikut:*

```
Algoritma Normalisasi_Fitur_MFCC

Input  : Dataset hasil cleaning dengan kolom MFCC₁–MFCC₁₃ dan
label_speaker

Output : Dataset hasil normalisasi dengan skala rata-rata 0 dan
simpangan baku 1
```

Langkah-langkah:

1. Membaca dataset hasil cleaning dari direktori penyimpanan.

2. Pilih kolom fitur utama ($MFCC_1$ sampai $MFCC_{13}$) sebagai input untuk normalisasi.

3. Terapkan metode StandardScaler (Z-Score Normalization) menggunakan rumus:

$$z_j = (x_j - \mu_j) / \sigma_j$$

di mana:

z_j = nilai fitur setelah normalisasi

x_j = nilai asli fitur ke-j

μ_j = rata-rata fitur ke-j

σ_j = simpangan baku fitur ke-j

4. Bentuk dataset baru yang berisi hasil normalisasi seluruh fitur MFCC dan kolom label asli.

5. Simpan hasil normalisasi dalam dua berkas:

- tabel10_sebelum_normalisasi.csv → data asli sebelum penskalaan

- tabel11_sesudah_normalisasi.csv → data setelah penskalaan

6. Tampilkan contoh nilai sebelum dan sesudah normalisasi untuk verifikasi.

Selesai

Tabel 4.4. Nilai Fitur Sebelum Normalisasi

Index_sample	MFCC_1	MFCC_2	MFCC_3	…	MFCC_13	Kelas
0	0.000000114	0.000000000	-0.000000009	…	0.000000000	Me
1	-0.000000418	0.000000057	-0.000000066	…	0.000000019	Me
2	0.000000304	0.000000076	-0.000000026	…	0.000000000	Me

Tabel 4.4 menunjukkan contoh hasil ekstraksi fitur MFCC sebelum dilakukan normalisasi, sementara Tabel 4.5 menampilkan hasil setelah normalisasi:

Tabel 4.5. Nilai Fitur Setelah Normalisasi

Index_sample	MFCC_1	MFCC_2	MFCC_3	…	MFCC_13	Kelas
0	0.4899	0.0236	−0.2616	…	−0.3355	Me
1	−1.8617	1.3753	−1.7760	…	−0.6099	Me
2	1.3297	−1.7786	−0.7033	…	−0.0206	Me

Nilai-nilai yang semula sangat kecil (di bawah 10^{-6}) telah ditransformasi ke skala standar (rata-rata 0 dan simpangan baku 1), sehingga setiap fitur memiliki kontribusi yang seimbang terhadap proses pelatihan model SVM.

4.2.4. Pengenalan Speaker dengan SVM

Support Vector Machine digunakan sebagai algoritma utama untuk melakukan pengenalan *speaker* antara dua kelas, yaitu *Me* (pengguna utama) dan *Notme* (penutur lain). SVM bekerja dengan membentuk sebuah fungsi keputusan (*decision function*) yang memaksimalkan jarak pemisah (*maximum margin*) antara dua kelas pada ruang fitur berdimensi tinggi.

Pada penelitian ini, setiap sampel suara direpresentasikan sebagai sebuah vektor fitur berdimensi 13 yang berasal dari koefisien $MFCC_1$ hingga $MFCC_{13}$. Vektor fitur tersebut dipetakan ke dalam ruang fitur SVM untuk membentuk *hyperplane* optimal yang memisahkan distribusi karakteristik suara kelas *Me* dan *Notme*. Selain menghasilkan label klasifikasi, SVM juga menghasilkan nilai keputusan (*decision score*) yang merepresentasikan jarak relatif suatu vektor fitur terhadap hyperplane pemisah. Nilai keputusan ini digunakan sebagai dasar penerapan mekanisme *threshold* untuk meningkatkan keandalan sistem dalam membedakan penutur sah dan tidak sah.

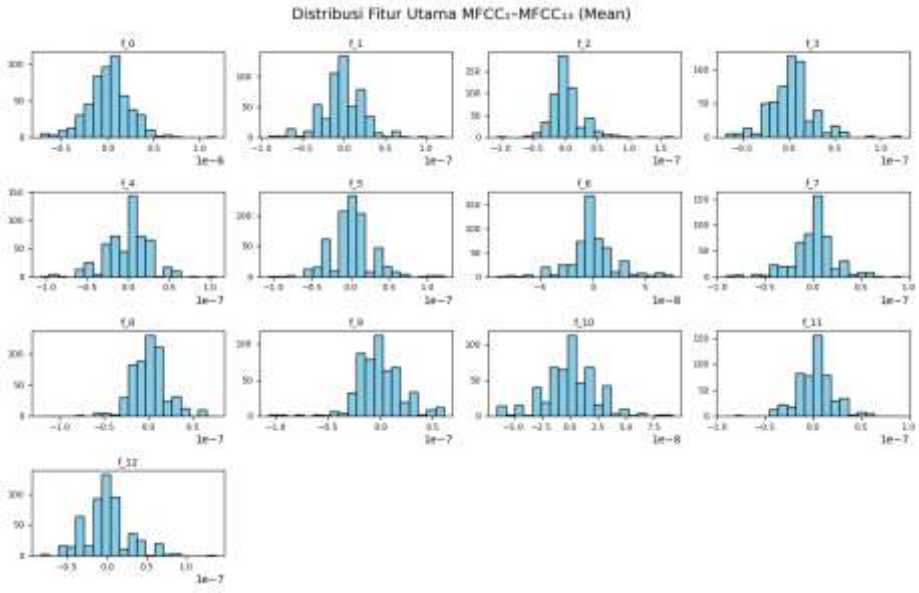
Penentuan nilai *threshold* dilakukan berdasarkan analisis *Receiver Operating Characteristic* (ROC), dengan memilih titik ambang pada *False Acceptance Rate* (FAR) sebesar 5%. Pemilihan FAR 5% bertujuan untuk menekan risiko penerimaan penutur tidak sah, mengingat sistem ini digunakan untuk mengontrol akses fisik pada pintu garasi.

Berdasarkan threshold tersebut, wilayah keputusan SVM dibagi menjadi tiga zona, yaitu *TH_high, TH_low*, dan zona ambigu. Jika nilai keputusan berada di atas *TH_high*, maka sampel suara diklasifikasikan sebagai *Me* dan sistem memberikan akses. Jika nilai keputusan berada di bawah *TH_low*, maka sampel suara diklasifikasikan sebagai *Notme* dan akses ditolak. Adapun nilai keputusan yang berada di antara *TH_low* dan *TH_high* dikategorikan sebagai kondisi ambigu, di mana sistem tidak langsung memberikan akses untuk meminimalkan kesalahan pengenalan.

Pendekatan dua ambang (*dual-threshold*) ini memungkinkan sistem mengontrol *trade-off* antara keamanan dan kenyamanan pengguna secara lebih fleksibel. Dengan adanya zona ambigu, sistem dapat menghindari keputusan yang bersifat tidak pasti, sehingga risiko kesalahan penerimaan (*false acceptance*) maupun kesalahan penolakan (*false rejection*) dapat ditekan secara lebih efektif.

4.2.4.1. Exploratory Data Analysis (EDA)

Exploratory Data Analysis dilakukan dengan memvisualisasikan histogram masing-masing fitur MFCC untuk memeriksa bentuk distribusi dan memastikan tidak terdapat nilai ekstrim atau pola yang tidak wajar.



Gambar 4.7. Histogram Fitur MFCC

Gambar 4.7 menunjukkan bahwa setiap fitur $MFCC_1$–$MFCC_{13}$ memiliki distribusi yang mendekati bentuk kurva normal dan berpusat di sekitar nol setelah proses normalisasi. Tidak ditemukan *outlier* ekstrem atau penyebaran nilai yang tidak simetris. Distribusi yang konsisten ini menandakan bahwa keseluruhan fitur berada dalam kondisi yang baik dan siap digunakan sebagai input bagi model SVM, yang sangat bergantung pada skala dan kestabilan fitur untuk menghasilkan hyperplane pemisah yang optimal.

4.2.4.2. Hasil Data Splitting

Proses pembagian data menghasilkan dua bagian, yaitu data latih dan data uji. Pembagian dilakukan secara acak terkontrol menggunakan parameter *random_state* dan *stratified split* untuk menjaga proporsi kelas pada data latih dan uji tetap konsisten. Tabel 4.6 menampilkan hasil pembagian data untuk tiga rasio yang berbeda.

Tabel 4.6. Hasil Rasio Pembagian Data

Skenario	Jumlah Data Latih	Jumlah Data Uji
90:10	486	54
80:20	432	108
70:30	378	162

4.2.4.3. Hasil Uji Coba Skenario 90:10

Pengujian pada skenario pembagian data 90:10 dilakukan untuk mengevaluasi performa model SVM setelah melalui tahap ekstraksi fitur MFCC dan normalisasi *Z-score*. Dari total 540 sampel, diperoleh 486 data latih dan 54 data uji, masing-masing tetap menjaga proporsi kelas *Me* dan *Notme* menggunakan *stratified split*. Distribusi data uji terdiri dari 27 sampel *Me* dan 27 sampel *Notme*, sehingga kedua kelas berada dalam kondisi seimbang. Kondisi ini memastikan bahwa proses evaluasi tidak dipengaruhi oleh bias distribusi kelas dan memberikan gambaran performa model secara objektif.

Pada pengujian ini, model SVM dievaluasi menggunakan dua jenis kernel, yaitu RBF dan Linear, dengan variasi hyperparameter *Cost* (C) dan *Gamma* (γ). Kernel RBF bekerja dengan memetakan data ke ruang berdimensi lebih tinggi menggunakan fungsi Gaussian, sehingga mampu menangkap hubungan non-linear antara fitur suara. Sebaliknya, kernel Linear berfokus pada pencarian *hyperplane* optimal yang secara linear memisahkan dua kelas data (*Me* dan *Notme*) dengan cara memaksimalkan margin antar *support vector*.

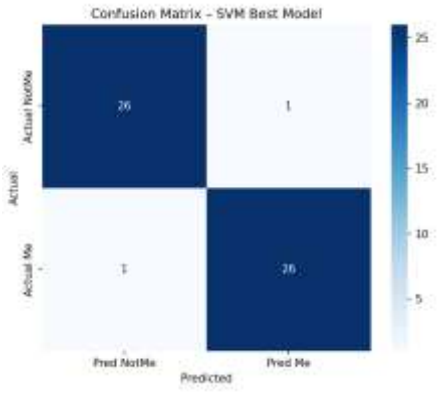
Proses tuning *hyperparameter* dilakukan secara *grid search* untuk mencari kombinasi C dan γ terbaik berdasarkan nilai *F1-Score*. Evaluasi performa dilakukan menggunakan *confusion matrix* dengan perhitungan metrik *Accuracy*, *Precision*,

Recall, dan *F1-Score*, yang menggambarkan kemampuan model dalam membedakan identitas antara *Me* dan *Notme*.

Tabel 4.7. Hasil Evaluasi Pengujian Skenario 1

Percobaan	Kernel	*Hyperparameter*		*Evaluasi*			
		Cost (C)	*Gamma (γ)*	*Accuracy (%)*	*Precision (%)*	*Recall (%)*	*F1-Score(%)*
1	RBF	0.1	0.1	92.6	92.59	92.59	92.59
2			0.5	90.75	90.74	90.74	90.74
3			1	90.75	90.74	90.74	90.74
4		1	0.1	96.3	96.15	96.3	96.22
5			0.5	96.3	96.15	96.3	96.22
6			1	96.3	96.15	96.3	96.22
7		10	0.1	96.3	3.7	96.15	96.3
8			0.5	96.3	3.70	96.30	96.30
9			1	94.45	5.55	94.44	94.44
10	Linear	0,1		96.3	96.15	96.3	96.22
11		1		96.3	96.15	96.3	96.22
12		10		96.3	96.15	96.3	96.22

Berdasarkan hasil evaluasi yang ditunjukkan pada Tabel 4.7, konfigurasi terbaik diperoleh pada model SVM dengan kernel RBF, nilai *Cost* (C) = 10, dan *Gamma* (γ) = 0.5. Kombinasi ini memberikan performa yang paling stabil dibandingkan konfigurasi lainnya, dengan nilai akurasi sebesar 96.30%, serta nilai *Precision, Recall,* dan *F1-Score* masing-masing sebesar 96.30%. Nilai akurasi yang tinggi ini menunjukkan bahwa model mampu mengklasifikasikan suara *Me* dan *Notme* secara konsisten dengan tingkat kesalahan yang rendah.



Gambar 4.8. *Confusion Matrix* Skenario 1

Hasil *confusion matrix* pada Gambar 4.8 menunjukkan bahwa model mampu mengklasifikasikan suara dengan sangat baik. Dari total 54 data uji, terdapat 26 sampel *Me* dan 26 sampel *Notme* yang berhasil diklasifikasikan dengan benar. Kesalahan hanya terjadi pada dua sampel, yaitu satu sampel *Notme* yang salah dikenali sebagai *Me* dan satu sampel *Me* yang salah dikenali sebagai *Notme*.

4.2.4.4. Hasil Uji Coba Skenario 80:20

Pengujian pada skenario pembagian data 80:20 dilakukan untuk mengevaluasi konsistensi performa model SVM saat porsi data latih dikurangi dan data uji diperbesar. Dari total 540 sampel, proses pembagian menghasilkan 432 data latih dan 108 data uji, dengan proporsi kelas tetap dijaga menggunakan *stratified split*. Distribusi data uji terdiri dari 54 sampel *Me* dan 54 sampel *Notme*, sehingga kedua kelas tetap berada dalam kondisi seimbang dan bebas bias distribusi.

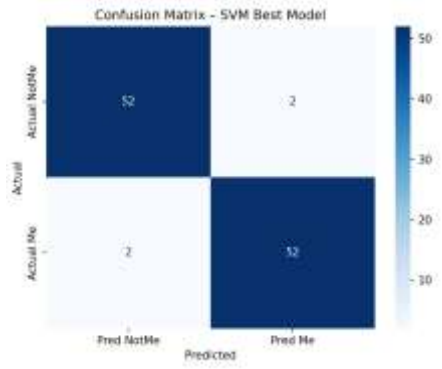
Seperti pada skenario sebelumnya, model SVM dievaluasi menggunakan dua jenis kernel, yaitu RBF dan Linear, yang masing-masing diuji dengan variasi hyperparameter *Cost* (C) dan *Gamma* (γ). Kernel RBF memetakan fitur ke ruang berdimensi tinggi untuk menangkap hubungan non-linear, sementara kernel Linear mencari garis batas pemisah terbaik pada ruang asli. Hasil pengujian seluruh kombinasi ditampilkan pada Tabel 4.8.

Tabel 4.8. Hasil Evaluasi Pengujian Skenario 2

Percobaan	Kernel	*Hyperparameter*		*Evaluasi*			
		Cost (C)	*Gamma (γ)*	*Accuracy (%)*	*Precision (%)*	*Recall (%)*	*F1-Score(%)*
1	RBF	0.1	0.1	92.59	92.59	92.59	92.59
2			0.5	90.74	90.74	90.74	90.74
3			1	90.74	90.74	90.74	90.74
4		1	0.1	96.3	96.15	96.3	96.22
5			0.5	96.3	96.15	96.3	96.22
6			1	96.3	96.15	96.3	96.22
7		10	0.1	96.3	96.15	96.3	96.22

8			0.5	96.3	96.3	96.3	96.3
9			1	94.45	94.44	94.44	94.44
10	Linear	0,1		96.3	96.15	96.3	96.22
11		1		96.3	96.15	96.3	96.22
12		10		96.3	96.15	96.3	96.22

Berdasarkan hasil evaluasi pada Tabel 4.8, konfigurasi terbaik diperoleh pada model SVM dengan kernel RBF, nilai *Cost* (C) = 10, dan *Gamma* (γ) = 0.1. Model ini memberikan performa paling optimal pada skenario pembagian data 80:20 dengan nilai akurasi sebesar 96.30%, serta nilai *Precision, Recall,* dan *F1-Score* masing-masing sebesar 96.30%. Konsistensi nilai evaluasi tersebut menunjukkan bahwa model tetap mampu mengenali pola suara *Me* dan *Notme* dengan baik meskipun proporsi data uji lebih besar.



Gambar 4.9. *Confusion Matrix* Skenario 2

Confusion matrix pada Gambar 4.9 menunjukkan bahwa model SVM dengan konfigurasi terbaik (kernel RBF, C = 10, γ = 0.1) mampu melakukan klasifikasi dengan tingkat akurasi yang sangat tinggi. Dari total 108 data uji, sebanyak 52 sampel *Notme* berhasil diklasifikasikan dengan benar sebagai *Notme*, dan 52 sampel *Me* juga terklasifikasi benar sebagai *Me*. Kesalahan prediksi hanya terjadi pada 4 sampel, yang terdiri dari 2 sampel *Notme* yang salah dikenali sebagai *Me*, serta 2 sampel *Me* yang salah diklasifikasikan sebagai *Notme*.

4.2.4.5. Hasil Uji Coba Skenario 70:30

Pengujian pada skenario pembagian data 70:30 dilakukan untuk mengukur kemampuan generalisasi model SVM ketika jumlah data uji tiga kali lebih besar dibandingkan skenario 90:10. Dari total 540 sampel, diperoleh 378 data latih dan 162 data uji menggunakan *stratified split*, sehingga proporsi kelas *Me* dan *Notme* tetap terjaga secara seimbang. Pada data uji, terdapat 81 sampel *Me* dan 81 sampel *Notme*, yang memastikan bahwa evaluasi performa model berlangsung tanpa bias distribusi kelas.

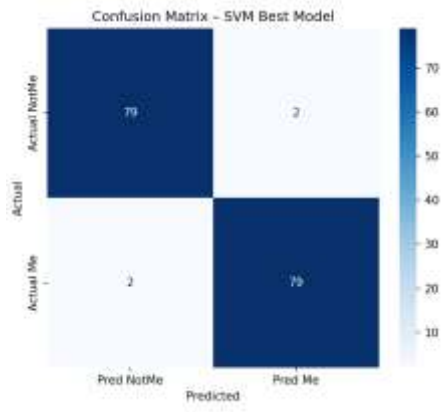
Pada pengujian ini, model SVM diuji menggunakan kernel RBF dan Linear dengan berbagai kombinasi hyperparameter *Cost* (C) dan *Gamma* (γ). Kernel RBF digunakan untuk menangani pola hubungan non-linear antar fitur suara, sedangkan kernel Linear digunakan untuk memisahkan kelas secara linear dengan margin optimal.

Tabel 4.9. Hasil Evaluasi Pengujian Skenario 3

Percobaan	**Kernel**	***Hyperparameter***		***Evaluasi***			
		Cost (C)	***Gamma (γ)***	***Accuracy (%)***	***Precision (%)***	***Recall (%)***	***F1-Score(%)***
1	RBF	0.1	0.1	90.12	90.12	90.12	90.12
2			0.5	88.89	88.89	88.89	88.89
3			1	88.89	88.89	88.89	88.89
4		1	0.1	95.06	95.06	95.06	95.06
5			0.5	95.06	95.06	95.06	95.06
6			1	95.06	95.06	95.06	95.06
7		10	0.1	96.3	96.3	96.3	96.3
8			0.5	97.53	98	98	98
9			1	93.83	93.83	93.83	93.83
10	Linear	0,1		95.06	95.06	95.06	95.06
11		1		95.06	95.06	95.06	95.06
12		10		95.06	95.06	95.06	95.06

Berdasarkan hasil evaluasi pada Tabel 4.9, konfigurasi terbaik diperoleh pada model SVM dengan kernel RBF, nilai *Cost* (C) = 10, dan *Gamma* (γ) = 0.5. Kombinasi ini menghasilkan nilai akurasi sebesar 97.53%, serta nilai *Precision*,

Recall, dan *F1-Score* masing-masing sebesar 98%. Hasil ini menunjukkan bahwa model mampu mempertahankan performa yang sangat stabil pada kedua kelas, dengan tingkat kesalahan yang rendah dalam membedakan suara pengguna (*Me*) dan penutur lain (*Notme*) pada skenario pembagian data 70:30.



Gambar 4.10. *Confusion Matrix* Skenario 3

Confusion matrix pada skenario 70:30 menampilkan hasil klasifikasi terhadap 162 data uji, yang terdiri dari 81 data *Me* dan 81 data *Notme*. Berdasarkan visualisasi pada Gambar 4.10, model menunjukkan performa yang sangat baik dengan tingkat kesalahan yang sangat rendah. Tercatat bahwa 79 sampel *Me* dan 79 sampel *Notme* berhasil diklasifikasikan dengan benar, sedangkan hanya terjadi empat kesalahan, yaitu masing-masing dua sampel Me yang salah dikenali sebagai *Notme*, dan dua sampel *Notme* yang salah dikenali sebagai *Me*.

4.2.5. Pencocokkan Kata dengan Regex

Pada tahap ini, sistem menggunakan regex sebagai mekanisme pengambilan keputusan deterministik berdasarkan teks hasil transkripsi dari modul *Speech-to-Text* (STT). Setelah sinyal suara melewati proses pra-pemrosesan serta dinyatakan sebagai pengguna sah (*Me*) oleh model SVM, sistem memanggil layanan *speech*

recognition untuk mendapatkan teks transkripsi. Dari teks inilah proses *intent detection* dilakukan menggunakan dua pola regex utama:

```
RE_BUKA  = \b(buka)\b
RE_TUTUP = \b(tutup)\b
```

Kedua pola ini bersifat *case-insensitive* dan menggunakan batas kata (\b) agar tidak salah mengenali kata serupa (misalnya "bukan" tidak dihitung sebagai "buka"). Setelah transkrip tersedia, sistem menelusuri seluruh kecocokan menggunakan fungsi *finditer*() untuk memperoleh posisi dan isi kata yang cocok, kemudian mengurutkannya berdasarkan urutan kemunculan dalam kalimat.

Keputusan intent dibuat secara deterministik dari urutan hit tersebut: jika token pertama yang cocok adalah "buka", *intent* ditetapkan buka, jika "tutup", *intent* menjadi tutup, dan jika tidak ada kecocokan sama sekali, sistem mengembalikan label *filler*. Perintah hanya dieksekusi jika dua kondisi terpenuhi bersamaan:

1) Pengguna terverifikasi sebagai *Me* oleh model SVM
2) Hasil regex termasuk salah satu perintah valid ("buka" atau "tutup").

Proses pencocokkan perintah dapat dilihat melalui *pseudocode* berikut:

```
Input : Teks hasil transkripsi dari modul Speech-to-Text (STT)
Output : Label perintah ("buka", "tutup", atau "filler")
Langkah-langkah:
   1. Inisialisasi dua pola regular expression (Regex):
         • RE_BUKA ← \b(buka)\b
         • RE_TUTUP ← \b(tutup)\b
   2. Terima input berupa hasil transkripsi teks.
         Jika teks kosong → kembalikan "filler".
```

3. Lakukan pencocokan terhadap kedua pola menggunakan finditer().

   Simpan setiap kecocokan ke dalam daftar matches dengan atribut kata dan posisi.

4. Urutkan daftar matches berdasarkan posisi kemunculan pertama.
5. Tentukan hasil deteksi:

   a. Jika ditemukan kata "buka" atau "open" → command ← "buka".

   b. Jika ditemukan kata "tutup" atau "close" → command ← "tutup".

   c. Jika tidak ditemukan keduanya → command ← "filler".

6. Kembalikan hasil deteksi (command, daftar kecocokan, dan kata terpilih).
7. Gunakan hasil ini bersama keluaran model SVM untuk menentukan tindakan akhir sistem.

Selesai

4.3. Implementasi Komunikasi IoT

Implementasi komunikasi IoT pada sistem ini menggunakan protokol MQTT dengan HiveMQ Cloud sebagai *broker*. HiveMQ (*broker* MQTT) adalah layanan perantara yang menerima pesan dari pihak yang menerbitkan (*publisher*), lalu merutekannya ke pihak yang berlangganan (*subscriber*) berdasarkan topik dengan dukungan koneksi aman (TLS), kontrol akses pengguna/topik, dan opsi QoS untuk menjamin pengantaran.

Gambar 4.11. Logo HiveMQ

Pada sisi PC yang menangani akuisisi suara, pemrosesan *machine learning*, dan klien MQTT, sistem menghasilkan keputusan perintah dalam bentuk *payload* JSON yang dikirim ke topik perintah pada broker. ESP32 sebagai perangkat pengendali berlangganan topik tersebut, membaca status pintu terakhir (buka/tutup), kemudian menentukan apakah diperlukan aksi berdasarkan indikator "flag". Aturannya tetap instruksi "buka" hanya dijalankan ketika pintu berada pada kondisi tertutup, dan instruksi "tutup" dieksekusi ketika pintu berada pada kondisi terbuka. Kasus *Notme* atau *filler* menghasilkan flag bernilai 0 sehingga tidak ada pergerakan. Perangkat kemudian mempublikasikan status singkat sebagai informasi real-time untuk pemantauan sistem.

Pemilihan QoS disesuaikan dengan kebutuhan operasional. Perintah dikirim menggunakan QoS 1 tanpa opsi *retain* untuk mencegah penyimpanan perintah lama pada *broker*. Status, event ringan, dan *telemetry* menggunakan QoS 0 agar beban jaringan tetap rendah. Seluruh keluaran *machine learning*, baik perintah valid maupun non-perintah seperti *filler* atau prediksi dengan kepercayaan rendah, dicatat dalam basis data sebagai log. *Telemetry* disimpan sebagai deret waktu, sedangkan status perangkat terkini diperbarui menggunakan mekanisme upsert agar selalu menampilkan kondisi terbaru. Setiap *payload* memuat penanda waktu dan `corr_id` (UUID) untuk memudahkan pelacakan dan mencegah terjadinya duplikasi aksi akibat pengiriman ulang. Koneksi menuju *broker* diamankan dengan TLS dan kredensial pengguna serta dapat dilengkapi fitur *keep-alive* dan *Last-Will* untuk mendeteksi putus sambungan secara otomatis.

4.4. Implementasi Sistem

Implementasi sistem pada penelitian ini menggabungkan aplikasi *mobile*, server *machine learning*, dan perangkat mikrokontroler ESP32. Aplikasi digunakan untuk merekam suara dan mengirimkannya ke server. Server kemudian memproses audio menggunakan model SVM dan Regex untuk menentukan identitas pengguna serta jenis perintah. Hasil keputusan dikirimkan melalui MQTT ke ESP32, yang selanjutnya menggerakkan motor servo, membaca sensor IR, dan menampilkan status melalui LED dan buzzer. Integrasi ini memungkinkan kontrol pintu garasi berjalan secara otomatis dan *real-time*.

4.4.1. Sistem *Software*

Sistem *software* pada penelitian ini berfungsi sebagai antarmuka pengguna (*user interface*) dan pengelola komunikasi data antara sistem pengenalan suara berbasis *machine learning* dengan perangkat mikrokontroler yang mengendalikan pintu garasi. Aplikasi ini dikembangkan menggunakan *framework* Flutter dengan arsitektur *client–server*, di mana aplikasi berperan sebagai *client* yang mengirimkan data suara pengguna ke server untuk diproses dan kemudian menerima hasil berupa identitas pengguna (*Me* atau *Notme*) serta perintah yang dikenali (*Buka*, *Tutup*, atau *Filler*).

Aplikasi ini juga terhubung dengan *broker* MQTT sebagai media komunikasi antara sistem *software* dan perangkat *hardware* ESP32. Pesan hasil pengenalan perintah suara dikirimkan melalui topik tertentu dan diteruskan ke mikrokontroler untuk mengeksekusi aksi fisik membuka atau menutup pintu garasi. Tampilan antarmuka aplikasi dirancang dengan prinsip sederhana dan fungsional agar

pengguna dapat dengan mudah merekam suara, mendaftarkan data suara baru, memantau aktivitas sistem, serta melihat analisis performa pengenalan suara.

Gambar 4.12. Tampilan *Voice Control*

Gambar 4.12 menunjukkan tampilan utama aplikasi *Smart Door Garage – Voice Control*. Halaman ini berfungsi sebagai antarmuka utama untuk melakukan perekaman suara pengguna. Pengguna dapat menekan ikon mikrofon di tengah layar untuk mulai merekam perintah suara seperti “Buka” atau “Tutup”. Setelah proses perekaman selesai, file audio akan dikirim ke server *backend* (*machine learning*) melalui protokol HTTP untuk diproses menggunakan model pengenalan suara berbasis SVM dan Regex. Hasil pemrosesan kemudian menentukan apakah suara tersebut dikenali sebagai pengguna sah (*Me*) dan perintah yang diucapkan valid.

Gambar 4.13. Tampilan *Analytics*

Gambar 4.13 menampilkan halaman *Analytics*, yang berfungsi untuk menyajikan hasil analisis aktivitas sistem secara visual. Halaman ini terdiri atas tiga bagian utama. Bagian pertama yaitu Statistik Aktivitas Pintu, ditampilkan dalam bentuk diagram batang yang memperlihatkan frekuensi perintah "Buka" dan "Tutup" yang diterima sistem. Bagian kedua adalah Deteksi *Speaker*, yang menunjukkan jumlah deteksi suara berdasarkan identitas pengguna, baik Speaker *Me* maupun *Speaker Notme*. Bagian ketiga yaitu Keberhasilan Deteksi dan Perintah, yang ditampilkan dalam bentuk diagram lingkaran (*pie chart*) untuk menggambarkan persentase keberhasilan sistem dalam mengenali kombinasi antara identitas pengguna dan jenis perintah yang benar. Fitur pada halaman ini membantu pengguna maupun pengembang dalam mengevaluasi performa sistem secara

keseluruhan, baik dari sisi efektivitas pengenalan suara maupun tingkat keberhasilan eksekusi perintah.

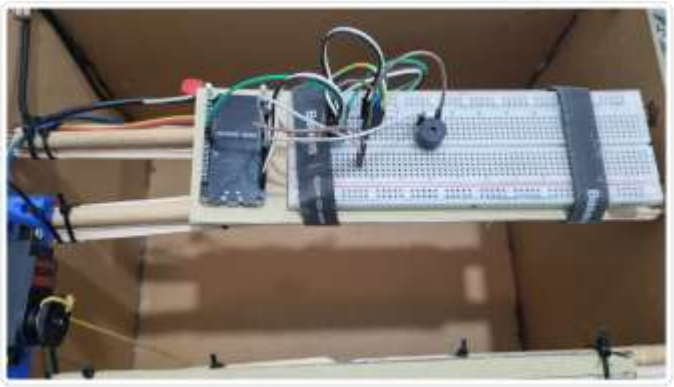
Gambar 4.14. Tampilan *Activity Logs*

Gambar 4.14 memperlihatkan halaman *Activity Logs*, yang berfungsi untuk menampilkan riwayat aktivitas sistem dalam mendeteksi perintah suara. Setiap entri log menampilkan jenis perintah yang diucapkan, identitas pengguna yang dikenali (*Me* atau *Notme*), serta waktu kejadian (*timestamp*) secara kronologis. Halaman ini memungkinkan pengguna untuk memantau jejak aktivitas sistem, termasuk perintah yang berhasil atau gagal dikenali, sehingga dapat digunakan untuk proses *debugging* dan evaluasi performa sistem secara real-time.

4.4.1. Sistem *Hardware*

Rangkaian *hardware* yang digunakan dalam Sistem Kontrol Pintu Garasi Pintar berbasis *Internet of Things* (IoT) ini terdiri dari beberapa komponen utama

yang saling terhubung untuk menjalankan fungsi kendali pintu garasi secara otomatis berdasarkan perintah suara. Sistem ini dirancang dengan menggunakan mikrokontroler ESP32 sebagai pusat pengendali, servo motor MG996R sebagai aktuator mekanik, sensor inframerah (IR *obstacle sensor*) untuk mendeteksi kondisi pintu, LED indikator sebagai penanda status sistem, dan buzzer sebagai alarm bunyi atau notifikasi proses.



Gambar 4.15. Rangkaian Mikrokontroler dengan Sensor

ESP32 berfungsi sebagai unit pengendali utama yang menerima perintah dari server backend melalui protokol MQTT. Setiap kali sistem menerima perintah suara dari pengguna yang dikenali sebagai “Buka” atau “Tutup”, data dikirim melalui broker MQTT ke ESP32. Mikrokontroler kemudian mengontrol servo motor MG996R yang dipasang di sisi kanan poros penggulung pintu. Servo ini berputar secara sinkron untuk menarik atau menurunkan daun pintu sesuai instruksi yang diterima.

Gambar 4.16. Rangkaian Motor Servo

Sensor IR ditempatkan di bagian bawah pintu untuk mendeteksi apakah pintu sudah tertutup sempurna. Sensor bekerja berdasarkan pantulan sinar inframerah ketika sinyal pantulan terhalang oleh pintu, sensor akan mengirimkan sinyal logika LOW ke ESP32, menandakan bahwa pintu telah menutup penuh.

Gambar 4.17. Penempatan IR Sensor

LED indikator berfungsi menunjukkan status sistem selama beroperasi. LED menyala stabil sebagai tanda bahwa perangkat berada dalam kondisi aktif, sedangkan mode kedip digunakan ketika servo sedang bergerak baik pada proses membuka maupun menutup pintu. Buzzer berfungsi sebagai indikator keamanan dan hanya berbunyi ketika sensor *infrared* mendeteksi adanya halangan saat pintu

sedang menutup. Bunyi berhenti ketika jalur kembali bersih dan servo melanjutkan proses penutupan. Seluruh komponen dirakit menggunakan breadboard dan dihubungkan melalui kabel jumper sesuai rancangan *pinout* ESP32. Sumber daya sistem berasal dari adaptor 5V DC yang menyalakan servo, sensor IR, LED, dan buzzer.

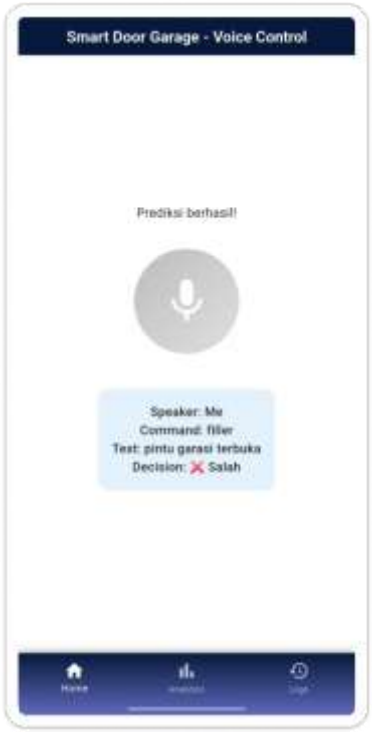
4.5. Hasil Pengujian Sistem

Pengujian sistem dilakukan untuk memastikan bahwa integrasi antara modul pengenalan suara, server *machine learning*, dan perangkat ESP32 dapat bekerja secara konsisten pada kondisi nyata.

4.5.1. Pengujian Sistem Kontrol

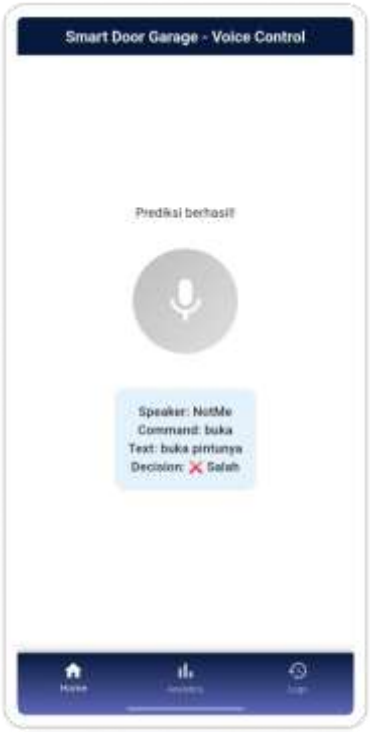
Pengujian sistem kontrol dilakukan untuk memastikan bahwa perangkat keras (*hardware*) dan perangkat lunak (*software*) dapat berfungsi secara terpadu dalam menjalankan proses pembukaan dan penutupan pintu garasi berdasarkan hasil pengenalan suara pengguna. Pengujian ini mencakup proses komunikasi antara ESP32 sebagai pengendali utama, server *machine learning* sebagai pemroses suara, dan *broker* MQTT sebagai media pertukaran data.

Seluruh perangkat diuji dalam kondisi terkoneksi ke jaringan *hotspot* eksternal yang berbeda dari jaringan pengembang untuk memastikan sistem dapat bekerja pada lingkungan jaringan yang bervariasi. ESP32 terhubung ke *broker* MQTT menggunakan protokol TLS (port 8883) untuk menjamin keamanan data, sedangkan server *backend* mengirimkan pesan perintah ke topik garasi/perintah berdasarkan hasil klasifikasi model.



Gambar 4.18. Suara Pengguna dengan Perintah Tidak Terdaftar

Gambar 4.18 menunjukkan hasil pengujian ketika pengguna sah (*Me*) memberikan perintah suara yang tidak terdaftar dalam sistem. Pada kondisi ini, sistem berhasil mengenali identitas suara pengguna sebagai *Me*, namun gagal mengenali perintah yang diucapkan karena tidak sesuai dengan pola Regex yang telah didefinisikan sebelumnya (yaitu “Buka” dan “Tutup”). Sebagai hasilnya, sistem tidak mengirimkan pesan perintah ke ESP32 dan servo motor tidak bergerak, yang menandakan bahwa fungsi validasi perintah bekerja dengan baik untuk mencegah eksekusi yang tidak sah.



Gambar 4.19. Suara bukan Pengguna yang sah

Gambar 4.19 menampilkan kondisi saat sistem menerima input suara dari pengguna yang tidak terdaftar (*Notme*). Berdasarkan hasil pengujian, sistem berhasil membedakan suara tersebut dari pengguna sah dengan akurasi tinggi. Model SVM mengklasifikasikan suara sebagai *Notme*, sehingga *backend* tidak meneruskan perintah apapun ke *broker* MQTT. Hal ini menunjukkan bahwa mekanisme otentikasi berbasis suara (*speaker verification*) bekerja dengan baik dalam menolak input dari pihak yang tidak berwenang, sehingga keamanan sistem dapat terjaga.

Gambar 4.20. Suara Pengguna dan Perintah Terdaftar

Gambar 4.20 memperlihatkan kondisi ketika pengguna sah (*Me*) mengucapkan perintah suara yang terdaftar dalam sistem, yaitu “Buka” atau “Tutup”. Dalam skenario ini, sistem berhasil mengenali identitas pengguna dan mencocokkan perintah melalui metode Regex. Setelah validasi berhasil, server mengirim pesan MQTT ke topik garasi/perintah, yang kemudian diterima oleh ESP32 untuk menggerakkan servo motor membuka atau menutup pintu sesuai perintah. Selain itu, LED indikator menyala dan buzzer berbunyi singkat sebagai tanda bahwa sistem telah mengeksekusi perintah dengan sukses. Respon sistem yang cepat dan akurat ini membuktikan bahwa integrasi antara pengenalan suara, backend, dan kontrol perangkat keras berjalan dengan baik.

4.5.2. Pengujian *Error* Sistem

Pengujian *error* sistem dilakukan untuk mengukur sejauh mana tingkat kesalahan yang terjadi dalam sistem dalam mengenali pengguna dan perintah suara yang diberikan oleh pengguna. Pengujian ini melibatkan 1 orang pengguna yang sah dan 6 orang pengguna yang tidak sah. Masing-masing pengguna diminta untuk memberikan 20 kali input perintah suara untuk sistem yang dibuat. Setiap input perintah suara dicatat dan dievaluasi untuk melihat sejauh mana sistem dapat mengenali perintah secara tepat atau terjadi *error* dalam pemrosesan suara tersebut. Berikut hasil pengujian *error* sistem yang peneliti tuangkan dalam beberapa tabel.

Tabel 4.10. Hasil Pengujian *Error* Sistem oleh Aulia

HASIL PENGUJIAN *ERROR* SISTEM				
Aulia				
No.	**Perintah**	**Hasil Perintah**	**Respon Sistem**	
			Merespon	**Tidak Merespon**
1.	“buka”	“buka”	✓	
2.	“tutup”	“tutup”		✓
3.	“tolong tutup pintunya”	“tolong tutup pintunya”		✓
4.	“tutup dong pintunya”	“tutup dong pintunya”		✓
5.	“cepat tutup pintunya”	“cepat tutup pintunya”		✓
6.	“tolong buka gerbangnya”	“tolong buka gerbangnya”	✓	
7.	“buka pintu sekarang”	“buka pintu sekarang”	✓	
8.	“woy tutup woy”	“woi tutup woi”		✓
9.	“buka pelan-pelan”	“buka pelan-pelan”		✓
10.	“tutup aja dulu”	“tutup aja dulu”		✓
11.	“ayo buka pintunya”	“ayo buka pintu”		✓
12.	“pintunya ditutup dong”	“pintunya ditutup dong”		✓
13.	“buka pintunya dong”	“buka pintunya dong”		✓
14,	“tolong buka cepat”	“tolong buka cepat”		✓
15.	“tutup sekarang”	“tutup sekarang”		✓
16.	“buka pintu depan”	“buka pintu depan”	✓	
17.	“buka cepetan”	“buka cepat”		✓
18.	“tutup pelan-pelan”	“tutup pelan-pelan”		✓
19.	“buka pintunya ya”	“buka pintunya ya”		✓
20.	“pintunya tolong tutup”	“pintunya tolong tutup”		✓
Total *Error* Sistem			**4**	

Tabel 4.11. Hasil Pengujian *Error* Sistem oleh Taufik

HASIL PENGUJIAN *ERROR* SISTEM				
Taufik				
No.	**Perintah**	**Hasil Perintah**	**Respon Sistem**	
			Merespon	**Tidak Merespon**
1.	“buka”	“buka”	✓	
2.	“tutup”	“tutup”	✓	
3.	“buka dong gerbangnya”	“buka dong gerbangnya”		✓
4.	“buka gerbangnya”	“buka gerbangnya”		✓
5.	“tutup gerbangnya dong”	“tutup gerbangnya dong”		✓
6.	“ayo buka cepat”	“ayo buka cepat”		✓
7.	“buka pintunya sekarang”	“buka pintunya sekarang”		✓
8.	“tutup dulu”	“tutup dulu”		✓
9.	“tolong buka pintunya”	“tolong buka pintunya”		✓
10.	“tutup rapat pintunya”	“tutup rapat pintunya”		✓
11.	“buka gerbang depan”	“buka gerbang depan”		✓
12.	“gerbang tolong buka”	“gerbang tolong buka”		✓
13.	“tutup cepat”	“tutup cepat”		✓
14,	“buka pintu rumah”	“buka pintu rumah”	✓	
15.	“tutup sekarang”	“tutup sekarang”		✓
16.	“buka pelan-pelan”	“buka pelan-pelan”		✓
17.	“tutup aja”	“tutup aja”		✓
18.	“buka pintunya dong”	“buka pintunya dong”		✓
19.	“tutup gerbang cepat”	“tutup gerbang cepat”		✓
20.	“buka pintu gerbang”	“buka pintu gerbang”		✓
Total *Error* Sistem			**3**	

Tabel 4.12. Hasil Pengujian *Error* Sistem oleh Awanda

HASIL PENGUJIAN *ERROR* SISTEM				
Awanda				
No.	**Perintah**	**Hasil Perintah**	**Respon Sistem**	
			Merespon	**Tidak Merespon**
1.	“buka”	“buka”	✓	
2.	“tutup”	“tutup”		✓
3.	“bre buka bre”	“free buka free”		✓
4.	“bukaaa”	“buka”		✓
5.	“woy buka woy”	“woi buka woi”		✓
6.	“cepat tutup pintu”	“cepat tutup pintu”		✓
7.	“buka dulu”	“buka dulu”		✓
8.	“tolong buka sekarang”	“tolong buka sekarang”		✓
9.	“tutup pelan”	“tutup pelan”		✓
10.	“buka dong”	“bukain dong”		✓
11.	“tutup aja”	“tutup aja”		✓
12.	“ayo buka pintunya”	“ayo buka pintunya”		✓
13.	“gerbangnya tutup”	“gerbangnya tutup”		✓

14,	“buka pintu depan”	“buka pintu depan”		✓
15.	“tutup cepat”	“tutup cepat”		✓
16.	“tolong tutup dulu”	“tolong tutup dulu”		✓
17.	“buka pintu pelan”	“buka pintu pelan”	✓	
18.	“buka gerbang”	“buka gerbang”		✓
19.	“tutup pintunya dong”	“tutup pintunya dong”		✓
20.	“buka sekarang”	“buka sekarang”		✓
Total *Error* Sistem			**2**	

Tabel 4.13. Hasil Pengujian *Error* Sistem oleh Satrio

HASIL PENGUJIAN *ERROR* SISTEM				
Satrio				
No.	**Perintah**	**Hasil Perintah**	**Respon Sistem**	
			Merespon	**Tidak Merespon**
1.	“buka”	“buka”	✓	
2.	“tutup”	“tutup”		✓
3.	“stop”	“stop”		✓
4.	“bang fik buka bang fik”	“bang fiq buka bank feed”		✓
5.	“le minta tolong bukakan”	“le minta tolong bukakan”		✓
6.	“buka dong”	“buka dong”		✓
7.	“tutup dulu”	“tutup dulu”		✓
8.	“buka pintu”	“buka pintu”		✓
9.	“tutup cepat”	“tutup cepat”	✓	
10.	“buka pelan-pelan”	“buka pelan-pelan”		✓
11.	“tolong buka”	“tolong buka”		✓
12.	“gerbang buka”	“gerbang buka”		✓
13.	“buka pintunya dong”	“buka pintunya dong”		✓
14,	“tutup aja dulu”	“tutup aja dulu”		✓
15.	“tolong tutup”	“tolong tutup”		✓
16.	“ayo buka sekarang”	“ayo buka sekarang”		✓
17.	“buka cepat”	“buka cepat”		✓
18.	“buka gerbang depan”	“buka gerbang depan”		✓
19.	“pintunya tolong tutup”	“pintunya tolong tutup”		✓
20.	“buka aja dulu”	“buka aja dulu”		✓
Total *Error* Sistem			**2**	

Tabel 4.14. Hasil Pengujian *Error* Sistem oleh Aditya

HASIL PENGUJIAN *ERROR* SISTEM				
Aditya				
No.	**Perintah**	**Hasil Perintah**	**Respon Sistem**	
			Merespon	**Tidak Merespon**
1.	“buka”	“buka”		
2.	“tutup”	“tutup”		✓

3.	“buka dong”	“bukain dong”		✓
4.	“tutup coy”	“tutup coy”		✓
5.	“tutupin dong”	“tutupin dong”		✓
6.	“ayo buka pintunya”	“ayo buka pintunya”	✓	
7.	“buka sekarang”	“buka sekarang”		✓
8.	“tolong tutup dulu”	“tolong tutup dulu”		✓
9.	“buka pintu gerbang”	“buka pintu gerbang”		✓
10.	“tutup pintu cepat”	“tutup pintu cepat”		✓
11.	“buka depan”	“buka depan”	✓	
12.	“tutup aja dulu”	“tutup aja dulu”		✓
13.	“tolong buka”	“tolong buka”		✓
14,	“buka cepat”	“buka cepat”		✓
15.	“gerbang tolong buka”	“gerbang tolong buka”		✓
16.	“tutup pelan”	“tutup pelan”		✓
17.	“buka pintunya dong”	“buka pintunya dong”		✓
18.	“buka aja”	“buka aja”	✓	
19.	“tutup sekarang”	“tutup sekarang”		✓
20.	“buka pelan-pelan”	“buka pelan-pelan”		✓
Total *Error* Sistem			**3**	

Tabel 4.15. Hasil Pengujian *Error* Sistem oleh Markamah

HASIL PENGUJIAN *ERROR* SISTEM				
Markamah				
No.	**Perintah**	**Hasil Perintah**	**Respon Sistem**	
			Merespon	**Tidak Merespon**
1.	“buka”	“buka”	✓	
2.	“tutup”	“tutup”		✓
3.	“tolong tutup pintunya”	“tolong tutup pintunya”		✓
4.	“tolong buka pintu”	“tolong buka pintu”		✓
5.	“bukain pintunya dong”	“bukain pintunya”		✓
6.	“buka pelan-pelan”	“buka pelan-pelan”		✓
7.	“tutup sekarang”	“tutup sekarang”		✓
8.	“ayo buka cepat”	“ayo buka cepat”		✓
9.	“tolong tutup dulu”	“tolong tutup dulu”		✓
10.	“buka aja dulu”	“buka aja dulu”	✓	
11.	“tutup pintu depan”	“tutup pintu depan”		✓
12.	“buka dong”	“buka dong”	✓	
13.	“tutup pelan”	“tutup pelan”		✓
14,	“gerbang tolong buka”	“gerbang tolong buka”		✓
15.	“buka pintunya cepat”	“buka pintunya cepat”		✓
16.	“tutup aja”	“tutup aja”		✓
17.	“buka pintu gerbang”	“buka pintu gerbang”		✓
18.	“tolong buka gerbang”	“tolong buka gerbang”		✓
19.	“tutup cepat”	“tutup cepat”		✓
20.	“buka sekarang”	“buka sekarang”		✓

Total *Error* Sistem	**3**

Tabel 4.16. Hasil Pengujian *Error* Sistem oleh Peneliti

HASIL PENGUJIAN *ERROR* SISTEM				
Peneliti				
No.	**Perintah**	**Hasil Perintah**	**Respon Sistem**	
			Merespon	**Tidak Merespon**
1.	“buka”	“buka”	✓	
2.	“tutup”	“tutup”	✓	
3.	“tolong buka pintunya”	“tolong buka pintunya”	✓	
4.	“tolong tutup pintunya”	“tolong tutup pintunya”	✓	
5.	“buka dong”	“buka dong”	✓	
6.	“tutup dong”	“tutup dong”		✓
7.	“buka pintu garasi”	“buka pintu garasi”	✓	
8.	“tutup garasi sekarang”	“tutup garasi sekarang”	✓	
9.	“ayo buka cepat”	“ayo buka cepat”	✓	
10.	“cepat tutup pintunya”	“cepat tutup pintu”	✓	
11.	“buka pelan-pelan”	“buka pelan-pelan”	✓	
12.	“tutup perlahan-lahan”	“tutup perlahan-lahan”	✓	
13.	“tolong segera buka pintu garasi”	“tolong segera buka pintu garasi”	✓	
14,	“tolong segera tutup pintu garasi”	“tolong segera tutup pintu garasi”	✓	
15.	“buka gerbang depan”	“buka gerbang depan”	✓	
16.	“tutup gerbang depan”	“tutup gerbang depan”	✓	
17.	“aku mau keluar, tolong buka pintunya”	“aku mau keluar, tolong buka pintunya”		✓
18.	“aku sudah sampai, tolong tutup gerbang”	“aku sudah sampai, tolong tutup gerbang”	✓	
19.	“hai, ini percobaan buka pintu”	“hai, ini percobaan buka”	✓	
20.	“hari ini uji coba tutup pintu garasi”	“hari ini uji coba tutup pintu garasi”	✓	
Total *Error* Sistem			**2**	

Berdasarkan hasil pengujian yang dilakukan sebanyak 140 kali oleh 7 pengguna, ditemukan bahwa *error* sistem terjadi sebanyak 19 kali. Tingkat *error* dapat dihitung menggunakan rumus berikut:

$$\mu = \frac{\sum Ei}{n} \times 100\%$$

$$\mu = \frac{19}{140} \times 100\%$$

$$\mu = 13.57\% \quad (4.4)$$

Hasil ini menunjukkan bahwa sistem umumnya responsif dan mampu mengeksekusi perintah suara dengan baik, namun masih terdapat sejumlah kondisi yang menyebabkan sistem gagal merespons secara tepat. Faktor penyebab utama error meliputi variasi kalimat perintah yang tidak tercakup dalam kamus perintah, seperti "tutup aja", "buka cepat", atau frasa bebas lain yang membuat regex tidak menemukan token *buka* atau *tutup*, kesalahan transkripsi dari modul *Speech-to-Text* (STT), terutama pada kata seru atau slang yang menghasilkan teks berbeda sehingga tidak cocok dengan pola regex, penolakan perintah karena skor verifikasi speaker berada di bawah *threshold* keamanan, menyebabkan beberapa suara sah dianggap bukan pengguna demi menghindari *false accept*, serta kualitas sinyal yang dipengaruhi cara pengguna berbicara, seperti terlalu cepat, terlalu pelan, atau adanya kebisingan yang mengacaukan ciri akustik MFCC sehingga sistem kesulitan mengenali *speaker*.

Selain itu, kondisi lingkungan seperti percakapan sekitar, gema ruangan, atau *noise* kecil dapat menyebabkan pergeseran fitur suara dan meningkatkan kemungkinan *false accept* maupun *false reject*, terutama apabila nada suara penutur lain mirip dengan pengguna sah. Untuk mengurangi *error* di masa mendatang, beberapa perbaikan yang dapat dilakukan meliputi memperluas kamus perintah agar lebih toleran terhadap variasi ujaran, menambahkan mekanisme *fallback* berbasis pencarian kata kunci utama ("buka", "tutup"), menambah data pelatihan dengan variasi gaya bicara dan intonasi, menerapkan *noise reduction* yang lebih

kuat, serta melakukan *training* dengan contoh audio ber-*noise* agar model lebih robust terhadap kondisi lingkungan nyata.

4.6. Integrasi Islam

Pemanfaatan teknologi dalam sistem garasi otomatis berbasis pengenalan suara dapat dianalisis melalui kerangka *Maqashid Syariah* sebagai prinsip etika dalam pengembangan ilmu pengetahuan. *Maqashid Syariah*, sebagaimana dijelaskan oleh para ulama ushul seperti al-Ghazali dan kemudian disistematisasi secara komprehensif oleh Imam al-Syatibi dalam *al-Muwāfaqāt*, bertujuan menjaga kemaslahatan manusia melalui lima unsur pokok *al-dharūriyyāt*: menjaga agama, jiwa, akal, keturunan, dan harta. Pembagian lima kebutuhan dasar ini dijelaskan secara eksplisit sebagai bagian dari *maslahah dharuriyyah* yang menjadi inti tujuan syariat (Kurniawan & Hudafi, 2021). Konsep tersebut dipertegas kembali oleh kajian kontemporer yang menyebutkan bahwa al-Syatibi menjadikan *maslahah* sebagai dasar utama dalam seluruh pemikiran ushul fiqhnya, sehingga Maqāshid Syariah berfungsi sebagai kerangka untuk memahami dan menilai manfaat suatu penerapan hukum maupun teknologi (Milhan, 2022).

Implementasi sistem dalam penelitian ini berkaitan langsung dengan tujuan *hifzh al-māl* melalui peningkatan keamanan harta. Sistem identifikasi suara yang hanya menerima perintah dari pemilik sah mampu meminimalkan risiko pencurian, akses tidak sah, dan potensi penyalahgunaan sehingga sejalan dengan prinsip perlindungan terhadap kepemilikan.

Aspek *hifzh an-nafs* tercermin dari kemampuan sistem meningkatkan keselamatan penghuni rumah. Pengamanan yang lebih selektif mengurangi potensi

ancaman fisik akibat upaya pembobolan atau manipulasi pintu garasi. Perlindungan terhadap jiwa termasuk bagian dari tujuan syariat yang menekankan pentingnya keselamatan manusia dalam setiap aktivitas dan pemanfaatan teknologi.

Aspek *hifzh al-'aql* terlihat melalui proses penelitian dan perancangan sistem yang memanfaatkan potensi akal sebagai instrumen untuk menghasilkan solusi yang bermanfaat. Pengolahan sinyal suara, penerapan *machine learning*, dan integrasi perangkat keras mencerminkan pemanfaatan akal yang selaras dengan etika keilmuan dalam Islam. Penggunaan teknologi sebagai hasil pemikiran ilmiah menjadi bagian dari pengembangan akal yang bernilai positif selama diarahkan pada kemaslahatan.

Aspek *hifzh an-nasl* terwujud melalui perlindungan privasi dan keamanan keluarga. Sistem yang mampu menolak perintah dari pihak yang tidak berwenang membantu menjaga ketenteraman rumah dan mengurangi risiko ancaman terhadap anggota keluarga. Keamanan ruang tempat tinggal termasuk bagian dari penjagaan keturunan dan kehormatan dalam perspektif syariat.

Aspek *hifzh ad-din* tercermin melalui penerapan nilai amanah dan tanggung jawab dalam pemanfaatan teknologi. Sistem yang dirancang dengan tujuan keamanan dan kemanfaatan menunjukkan upaya menjaga etika penggunaan teknologi agar tidak menimbulkan kerusakan. Prinsip amanah menjadi dasar moral dalam pengembangan sistem yang memberikan manfaat dan mengurangi potensi mudarat.

Integrasi kelima aspek *Maqāshid Syariah* tersebut menunjukkan bahwa penelitian ini tidak hanya menghasilkan inovasi teknologi, tetapi juga memenuhi

prinsip kemaslahatan dalam Islam. Sistem garasi otomatis berbasis pengenalan suara yang dikembangkan mencerminkan upaya pemanfaatan teknologi secara bertanggung jawab dengan tetap memperhatikan nilai-nilai *Maqāshid syariah* yang relevan bagi kehidupan modern.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan implementasi dan pengujian yang dilakukan, sistem Kontrol Pintu Garasi berbasis pengenalan suara menggunakan SVM dan Regex telah bekerja dengan baik dalam mengenali identitas speaker (Me dan Notme) serta perintah "buka" dan "tutup". Dari tiga skenario pembagian data (90:10, 80:20, dan 70:30), model SVM dengan kernel RBF secara konsisten memberikan performa terbaik pada skenario 70:30, dengan akurasi mencapai 97.53% serta nilai *Precision*, *Recall*, dan *F1-Score* yang sama-sama mencapai 98%. Hasil ini menunjukkan kemampuan generalisasi model yang kuat dalam mengenali suara pengguna. Arsitektur sistem mendukung integrasi penuh melalui penggunaan protokol HTTP sebagai jalur komunikasi antara aplikasi *mobile* dan *backend machine learning*, sehingga setiap rekaman perintah dapat dikirim, diproses, dan dikembalikan hasil prediksinya secara terstruktur. Integrasi tersebut dihubungkan dengan pencocokan Regex dan komunikasi MQTT untuk melakukan eksekusi perintah secara *real time* pada ESP32. Meskipun demikian, masih ditemukan *error* sistem sebesar 13.57%, terutama disebabkan oleh variasi kalimat, kesalahan transkripsi, serta kemiripan karakteristik suara, sehingga dibutuhkan perluasan dataset dan penyempurnaan mekanisme deteksi agar sistem semakin robust di kondisi nyata.

5.2 Saran

Penelitian ini masih dapat dikembangkan lebih lanjut, terutama dalam hal peningkatan variasi data dan metode klasifikasi. Disarankan agar jumlah partisipan dan kondisi lingkungan diperluas agar model dapat beradaptasi lebih baik terhadap berbagai karakter suara dan tingkat kebisingan. Penggunaan metode deep learning seperti CNN atau RNN dapat dipertimbangkan untuk meningkatkan akurasi pengenalan suara. Selain itu, sistem dapat ditingkatkan dari sisi keamanan komunikasi MQTT dengan menambahkan autentikasi dan enkripsi, serta dikembangkan menjadi aplikasi mobile dengan fitur notifikasi real-time agar pengguna dapat memantau pintu garasi dari jarak jauh.

Dari sisi perangkat keras, pengembangan selanjutnya dapat mempertimbangkan penggunaan sensor jarak yang lebih presisi. Pada penelitian ini, sensor *infrared* digunakan sebagai pendeteksi keberadaan objek dengan keluaran logika biner (terdeteksi atau tidak terdeteksi), sehingga jarak deteksi tidak dapat ditentukan secara kuantitatif. Untuk aplikasi yang membutuhkan pengukuran jarak dalam satuan tertentu, sensor ultrasonik dapat digunakan sebagai alternatif karena mampu memberikan nilai jarak secara numerik dan lebih fleksibel dalam penentuan ambang batas deteksi.

DAFTAR PUSTAKA

Adler, J., Azhar, M., & Supatmi, S. (2013). *Identifikasi Suara dengan MATLAB sebagai Aplikasi Jaringan Syaraf Tiruan*. *1*(1).

Adnan, F., & Amelia, I. (2022). *Implementasi Voice Recognition Berbasis Machine Learning*. *11*.

Ajmera, P. (2017). A Review Paper on Infrared sensor. *International Journal of Engineering Research*, *5*(23).

Akrimni, I. D., Akbi, D. R., & Sari, Z. (2024). Rancang Bangun Pintu Otomatis Berbasis Arduino RFID dan Voice Recognition Arduino. *Jurnal Repositor*, *6*(1). https://doi.org/10.22219/repositor.v6i1.31073

As Sarofi, M. A., Irhamah, I., & Mukarromah, A. (2020). Identifikasi Genre Musik dengan Menggunakan Metode Random Forest. *Jurnal Sains dan Seni ITS*, *9*(1). https://doi.org/10.12962/j23373520.v9i1.51311

Asbari, M., Jum'a, H., & Wulandari, Y. (n.d.). *Ketika Dakwah Bertemu Teknologi: Ikhtiar Menjaga Akal Sehat Di Era Digital*.

B. Davis, S., & Mermelstein, P. (1980). *Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences*. *4*. https://courses.physics.illinois.edu/ece417/fa2017/

Bahar, B., & Daniel Yc. Raban, R. (2023). Penerapan Algoritme Regular Expression Dalam Aplikasi Pendeteksi Nominal Uang Kertas. *Progresif: Jurnal Ilmiah Komputer*, *19*(2), 725. https://doi.org/10.35889/progresif.v19i2.1518

Darmawan, I. (2017). *Sound Film*. Pusat Pengembangan Perfilman Kementerian Pendidikan dan Kebudayaan.

Eray, O., Tokat, S., & Iplikci, S. (2018). An application of speech recognition with support vector machines. *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, 1–6. https://doi.org/10.1109/ISDFS.2018.8355321

Ermawati, Y., Purba, A., Despa, D., Z, F., & P, I. (2023). Prototype Pengontrol Pintu Garasi Rumah Dengan Motor Stepper Berbasis Arduino Menggunakan Perintah Suara. *Seminar Nasional Insinyur Profesional (SNIP)*, *3*(2). https://doi.org/10.23960/snip.v3i2.460

Fathur Rizky, R., Turmudi Zy, A., & Sunge, A. S. (2023). Sistem Smart Door Lock Menggunakan Voice Recognition Berbasis Arduino. *Bulletin of Information Technology (BIT)*, *4*(2), 239–244. https://doi.org/10.47065/bit.v4i2.696

Gourisaria, M. K., Agrawal, R., Sahni, M., & Singh, P. K. (2024). Comparative analysis of audio classification with MFCC and STFT features using machine learning techniques. *Discover Internet of Things*, *4*(1), 1. https://doi.org/10.1007/s43926-023-00049-y

Hanani, A., & Hariyadi, M. A. (2020). Smart Home Berbasis IoT Menggunakan Suara Pada Google Assistant. *Jurnal Ilmiah Teknologi Informasi Asia*, *14*(1), 49–56. https://doi.org/10.32815/jitika.v14i1.456

Handoko, R. B., & Suyanto, S. (2019). Klasifikasi Gender Berdasarkan Suara Menggunakan Support Vector Machine. *Indonesian Journal on Computing (Indo-JC)*, *4*(1), 9. https://doi.org/10.21108/INDOJC.2019.4.1.244

Hercog, D., Lerher, T., Truntič, M., & Težak, O. (2023). Design and Implementation of ESP32-Based IoT Devices. *Sensors*, *23*(15), 6739. https://doi.org/10.3390/s23156739

Heriyanto, H., Hartati, S., & Putra, A. E. (2018). EKSTRAKSI CIRI MEL FREQUENCY CEPSTRAL COEFFICIENT (MFCC) DAN RERATA COEFFICIENT UNTUK PENGECEKAN BACAAN AL-QUR'AN. *Telematika*, *15*(2), 99. https://doi.org/10.31315/telematika.v15i2.3123

Hutasoit, M. (2021). Rancang Bangun Prototipe Pengontrol Pintu Garasi Berbasis Internet of Things (IoT) Dengan Platform Android. *EINSTEIN*, *9*(1), 1. https://doi.org/10.24114/eins.v9i1.33760

Irugalbandara, C., Naseem, A. S., Perera, S., Kiruthikan, S., & Logeeshan, V. (2023). A Secure and Smart Home Automation System with Speech Recognition and Power Measurement Capabilities. *Sensors*, *23*(13), 5784. https://doi.org/10.3390/s23135784

Jefiza, A., Puspita, W. R., Firdaus, F., Sihombing, F. A., Setiawan, A., Herlina, G., Milala, G. I. L. S., & Fernanda, H. (2024). Smart home: Pintu Otomatis Berbasis Voice Recognition. *JURNAL INTEGRASI*, *16*(1), 62–67. https://doi.org/10.30871/ji.v16i1.7400

Jun, Z. (2021). The Development and Application of Support Vector Machine. *Journal of Physics: Conference Series*, *1748*(5), 052006. https://doi.org/10.1088/1742-6596/1748/5/052006

Kasaraneni, P. P., Venkata Pavan Kumar, Y., Moganti, G. L. K., & Kannan, R. (2022). Machine Learning-Based Ensemble Classifiers for Anomaly Handling in Smart Home Energy Consumption Data. *Sensors*, *22*(23), 9323. https://doi.org/10.3390/s22239323

Khadar Nawas, K., Kumar Barik, M., & Nayeemulla Khan, A. (2021). Speaker Recognition using Random Forest. *ITM Web of Conferences*, *37*, 01022. https://doi.org/10.1051/itmconf/20213701022

Kinnunen, T., & Li, H. (2010). An overview of text-independent speaker recognition: From features to supervectors. *Speech Communication*, *52*(1), 12–40. https://doi.org/10.1016/j.specom.2009.08.009

Kurniawan, A., & Hudafi, H. (2021). *KONSEP MAQASHID SYARIAH IMAM ASY-SYATIBI DALAM KITAB AL-MUWAFAQAT*. *15*(1).

Marini, M., Vanello, N., & Fanucci, L. (2021). Optimising Speaker-Dependent Feature Extraction Parameters to Improve Automatic Speech Recognition Performance for People with Dysarthria. *Sensors*, *21*(19), 6460. https://doi.org/10.3390/s21196460

Martin, R. S., Dewanto, D. Y., & Suryadarma, U. M. (2023). *PROTOTIPE KUNCI PINTU OTOMATIS MENGGUNAKANSENSOR KAMERA BERBASIS RASPBERRY*. *12*.

Megawati, S. (2021). Pengembangan Sistem Teknologi Internet of Things Yang Perlu Dikembangkan Negara Indonesia. *Journal of Information*

Engineering and Educational Technology, *5*(1), 19–26. https://doi.org/10.26740/jieet.v5n1.p19-26

Milhan, M. (2022). MAQASHID SYARI'AH MENURUT IMAM SYATIBI DAN DASAR TEORI PEMBENTUKANNYA. *Al-Usrah : Jurnal Al Ahwal As Syakhsiyah*, *9*(2). https://doi.org/10.30821/al-usrah.v9i2.12335

Mohamed Ariff, M. I., Mohamad Fadzir, F. D., Arshad, N. I., Ahmad, S., Salleh, K. A., & Wahab, J. A. (2022). Design and Development of a smart garage door system. *2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, 1–6. https://doi.org/10.1109/IEMTRONICS55184.2022.9795768

Nang An, N., Thanh Trung, T., Kim Hoan, T., Tuan Anh, N., & Minh Doanh, P. (2023). Integrate deep neural network and support vector machine to improve the quality of voice processing in internet of things devices. *Vinh University Journal of Science*, *52*(1A), 5–16. https://doi.org/10.56824/vujs.2023a005

Plaza R, M. A. J. & Hanum Maghfiro Risky Ningtias. (2023). PENERAPAN INTERNET OF THINGS PADA PROTOTYPE SMART HOME MENGGUNAKAN POLA SUARA DENGAN MIKROKONTROLER NODEMCU. *Sienna*, *4*(2), 87–96. https://doi.org/10.47637/sienna.v4i2.981

Prafanto, A., Budiman, E., Widagdo, P. P., Putra, G. M., & Wardhana, R. (2021). Pendeteksi Kehadiran menggunakan ESP32 untuk Sistem Pengunci Pintu Otomatis. *JTT (Jurnal Teknologi Terapan)*, *7*(1), 37. https://doi.org/10.31884/jtt.v7i1.318

Prayetno, M. A. (2022). Automatic Garage Door Prototype Using Arduino UNO-Based Sound Sensor. *JTECS : Jurnal Sistem Telekomunikasi Elektronika Sistem Kontrol Power Sistem dan Komputer*, *2*(1), 37. https://doi.org/10.32503/jtecs.v2i1.2299

Ramba, L. S., & Aria, M. (2020). Design Of A Voice Controlled Home Automation System Using Deep Learning Convolutional Neural Network (DL-CNN). *Telekontran : Jurnal Ilmiah Telekomunikasi, Kendali dan Elektronika Terapan*, *8*(1), 57–73. https://doi.org/10.34010/telekontran.v8i1.3078

Ridwansyah, M. A., & Rizal, A. (n.d.). *DESIGN OF SMART DOOR KEY SYSTEM BASED ON VOICE RECOGNITION USING MEL FREQUENCY CEPSTRAL COEFFICIENT (MFCC) AND K-NEAREST NEIGHBOR (K-NN)*.

Sari, I. P., Hazidar, A. H., Basri, M., Ramadhani, F., & Manurung, A. A. (2023). Penerapan Palang Pintu Otomatis Jarak Jauh Berbasis RFID di Perumahan. *Blend Sains Jurnal Teknik*, *2*(1), 16–25. https://doi.org/10.56211/blendsains.v2i1.246

Sasongko, S. M. A., Tsaury, S., Ariessaputra, S., & Ch, S. (2023). *Mel Frequency Cepstral Coefficients (MFCC) Method and Multiple Adaline Neural Network Model for Speaker Identification*.

Susanto, F., Prasiani, N. K., & Darmawan, P. (2022). IMPLEMENTASI INTERNET OF THINGS DALAM KEHIDUPAN SEHARI-HARI. *Jurnal Imagine*, *2*(1), 35–40. https://doi.org/10.35886/imagine.v2i1.329

Swastika, W., Widodo, R. B., & Oepojo, A. A. (2023). Perbandingan Akurasi Deteksi Emosi Pada Suara Menggunakan Multilayer Perceptron, Random Forest, Decision Tree dan K-NN. *Journal of Intelligent System and Computation*, *5*(1), 17–22. https://doi.org/10.52985/insyst.v5i1.264

Utama, S. N., Prakasa, J. E. W., & Hariyanto, W. (2025). *Klasifikasi Irama Bacaan Al-Qur'an Menggunakan Algoritma CNN*. *7*(1).

Zabar, A. A., & Novianto, F. (2015). KEAMANAN HTTP DAN HTTPS BERBASIS WEB MENGGUNAKAN SISTEM OPERASI KALI LINUX. *Komputa : Jurnal Ilmiah Komputer dan Informatika*, *4*(2), 69–74. https://doi.org/10.34010/komputa.v4i2.2427

Zhu, G., Caceres, J.-P., & Salamon, J. (2022). Filler Word Detection and Classification: A Dataset and Benchmark. *Interspeech 2022*, 3769–3773. https://doi.org/10.21437/Interspeech.2022-10992

LAMPIRAN

Lampiran 1. Video Demonstrasi Sistem

Video Demonstrasi dapat diakses melalui tautan berikut:
https://bit.ly/DemoSDG

Lampiran 2. Lembar Pengujian *Error* Sistem

PENGUJIAN ERROR SISTEM
SISTEM PINTU GARASI PINTAR MENGGUNAKAN VOICE RECOGNITION

Nama Responden : Awando
Tanggal Pengujian : 13 November 2025

HASIL PENGUJIAN ERROR SISTEM

No.	Perintah	Hasil Perintah	Respon Sistem		Catatan
			Merespon	Tidak Merespon	
1.	"buka"	"buka"	✓		
2.	"tutup"	"tutup"		✓	
3.	"bre buka bre"	"bre buka bre"		✓	
4.	"bukaaa"	"buka"		✓	
5.	"woy buka woy"	"woi buka woi"		✓	
6.	"cepat tutup pintu"	"cepat tutup pintu"		✓	
7.	"buka dulu"	"buka dulu"		✓	
8.	"tolong buka sekarang"	"tolong buka sekarang"		✓	
9.	"tutup pelan"	"tutup pelan"		✓	
10.	"buka dong"	"buka dong"		✓	
11.	"tutup aja"	"tutup aja"		✓	
12.	"ayo buka pintunya"	"ayo buka pintu"		✓	
13.	"gerbangnya tutup"	"gerbangnya tutup"		✓	
14.	"buka pintu depan"	"buka pintu depan"		✓	
15.	"tutup cepat"	"tutup cepat"		✓	
16.	"tolong tutup pintu"	"tolong tutup pintu"		✓	
17.	"buka pintu pelan"	"buka pintu pelan"	✓		
18.	"buka gerbang"	"buka gerbang"		✓	
19.	"tutup pintunya dong"	"tutup pintu dong"		✓	
20.	"buka sekarang"	"buka sekarang"		✓	
Total Error Sistem			2		

PENGUJIAN ERROR SISTEM
SISTEM PINTU GARASI PINTAR MENGGUNAKAN VOICE RECOGNITION

Nama Responden : Salmo
Tanggal Pengujian : 13 November 2025

HASIL PENGUJIAN ERROR SISTEM					
No.	Perintah	Hasil Perintah	Respon Sistem		Catatan
			Merespon	Tidak Merespon	
1.	"buka"	"buka"	✓		
2.	"tutup"	"tutup"		✓	
3.	"stop"	"stop"		✓	
4.	"bang pik buka bang pik"	"bang pig buka bank pred"		✓	
5.	"ic minta tolong bukakan"	"ic minta tolong bukakan"		✓	
6.	"buka dong"	"buka dong"		✓	
7.	"tutup dulu"	"tutup dulu"		✓	
8.	"buka pintu"	"buka pintu"		✓	
9.	"tutup cepat"	"tutup cepat"		✓	
10.	"buka pelan-pelan"	"buka pelan"	✓		
11.	"tolong buka"	"tolong buka"		✓	
12.	"gerbang buka"	"gerbang buka"		✓	
13.	"buka pintunya dong"	"buka pintunya dong"		✓	
14.	"tutup aja dulu"	"tutup aja dulu"		✓	
15.	"tolong tutup"	"tolong tutup"		✓	
16.	"aya buka sekarang"	"aya buka sekarang"		✓	
17.	"buka cepat"	"buka cepat"		✓	
18.	"buka gerbang"	"buka gerbang"		✓	
19.	"pintunya tolong tutup"	"pintu tolong tutup"		✓	
20.	"buka aja dulu"	"buka aja dulu"		✓	
Total Error Sistem			2		

PENGUJIAN ERROR SISTEM
SISTEM PINTU GARASI PINTAR MENGGUNAKAN VOICE RECOGNITION

Nama Responden : Adityo
Tanggal Pengujian : 13 November 2025

HASIL PENGUJIAN ERROR SISTEM					
No.	Perintah	Hasil Perintah	Respon Sistem		Catatan
			Merespon	Tidak Merespon	
1.	"buka"	"buka"		✓	
2.	"tutup"	"tutup"		✓	
3.	"buka dong"	"buka dong"		✓	
4.	"tutup coy"	"tutup coy"		✓	
5.	"tutupin dong"	"tutupin dong"		✓	
6.	"ayo buka pintunya"	"ayo buka pintu"	✓		
7.	"buka sekarang"	"buka sekarang"		✓	
8.	"tolong tutup pintu"	"tolong tutup pintu"		✓	
9.	"buka depan"	"buka depan"		✓	
10.	"tutup pintu cepat"	"tutup"		✓	
11.	"buka depan"	"buka depan"	✓		
12.	"tutup aja dulu"	"tutup aja dulu"		✓	
13.	"tolong buka"	"tolong buka"		✓	
14.	"buka cepat"	"buka cepat		✓	
15.	"gerbang tolong buka"	"gerbang tolong buka"		✓	
16.	"tutup pelan"	"tutup pelan"		✓	
17.	"buka pintunya dong"	"buka pintunya dong"		✓	
18.	"buka aja"	"buka aja"	✓		
19.	"tutup sekarang"	"tutup sekarang"		✓	
20.	"buka pelan-pelan"	"buka pelan-pelan"		✓	
Total Error Sistem			3		

PENGUJIAN ERROR SISTEM
SISTEM PINTU GARASI PINTAR MENGGUNAKAN VOICE RECOGNITION

Nama Responden : Marlenah
Tanggal Pengujian : 13 November 2023

HASIL PENGUJIAN ERROR SISTEM					
No.	Perintah	Hasil Perintah	Respon Sistem		Catatan
			Merespon	Tidak Merespon	
1.	"buka"	"buka"	✓		
2.	"tutup"	"tutup"		✓	
3.	"tolong tutup pintunya"	"tolong tutup pintu"		✓	
4.	"bukain pintunya dong"	"bukain pintu"		✓	
5.	"tolong buka pintu"	"tolong buka pintu"		✓	
6.	"buka pelan-pelan"	"buka pelan-pelan"		✓	
7.	"tutup sekarang"	"tutup sekarang"		✓	
8.	"ayo buka cepat"	"ayo buka cepat"		✓	
9.	"tolong tutup dulu"	"tolong tutup dulu"		✓	
10.	"buka aja dulu"	"buka aja dulu"	✓		
11.	"tutup pintu depan"	"tutup pintu depan"		✓	
12.	"buka dong"	"buka dong"	✓		
13.	"tutup pelan"	"tutup pelan"		✓	
14.	"gerbang tolong buka"	"gerbang tolong buka"		✓	
15.	"buka pintunya cepat"	"buka pintu"		✓	
16.	"tutup aja"	"tutup aja"		✓	
17.	"buka pintu gerbang"	"buka pintu gerbang"		✓	
18.	"tolong buka gerbang"	"tolong buka gerbang"		✓	
19.	"tutup cepat"	"tutup cepat"		✓	
20.	"buka sekarang"	"buka sekarang"		✓	
Total Error Sistem			3		

PENGUJIAN ERROR SISTEM
SISTEM PINTU GARASI PINTAR MENGGUNAKAN VOICE RECOGNITION

Nama Responden : Aulia

Tanggal Pengujian : 13 November 2025

HASIL PENGUJIAN ERROR SISTEM					
No.	Perintah	Hasil Perintah	Respon Sistem		Catatan
			Merespon	Tidak Merespon	
1.	"buka"	"buka"	V		
2.	"tutup"	"tutup"		V	
3.	"tolong tutup pintunya"	"tolong tutup pintunya"		V	
4.	"tutup dong pintunya"	"tutup dong pintunya"		V	
5.	"cepat tutup pintunya"	"cepat tutup pintunya"		V	
6.	"tolong buka gerbangnya"	"tolong buka gerbangnya"	V		
7.	"buka pintu sekarang"	"buka pintu sekarang	V		
8.	"oey tutup oey"	"woi tutup woi"		V	
9.	"buka pelan-pelan"	"buka pelan-pelan"		V	
10.	"tutup aja dulu"	"tutup aja dulu"		V	
11.	"ayo buka pintunya"	"ayo buka pintunya"		V	
12.	"pintunya ditutup dong"	"pintunya ditutup dong"		V	
13.	"buka pintunya dong"	"buka pintunya dong"		V	
14.	"tolong buka cepat"	"tolong buka cepat"		V	
15.	"tutup sekarang"	"tutup sekarang"		V	
16.	"buka pintu depan"	"buka pintu"	V		
17.	"buka cepetan"	"buka cepat"		V	
18.	"tutup pelan-pelan"	"tutup pelan-pelan"		V	
19.	"buka pintunya ya"	"buka pintunya ya"		V	
20.	"pintunya tolong tutup"	"pintunya tolong tutup"		V	
Total Error Sistem			A		

PENGUJIAN ERROR SISTEM
SISTEM PINTU GARASI PINTAR MENGGUNAKAN VOICE RECOGNITION

Nama Responden : Taufik
Tanggal Pengujian : 13 November 2025

HASIL PENGUJIAN ERROR SISTEM					
No.	Perintah	Hasil Perintah	Respon Sistem		Catatan
			Merespon	Tidak Merespon	
1.	"buka"	"buka"	V		
2.	"tutup"	"tutup"	V		
3.	"buka dong gerbangnya"	"buka dong gerbangnya"		V	
4.	"buka gerbangnya"	"buka gerbang"		V	
5.	"tutup gerbangnya dong"	"tutup gerbangnya dong"		V	
6.	"ayo buka cepat"	"ayo buka cepat"		V	
7.	"buka pintunya sekarang"	"buka pintunya sekarang"		V	
8.	"tutup dulu"	"tutup dulu"		V	
9.	"tolong buka pintunya"	"tolong buka pintunya"		V	
10.	"tutup rapat pintunya"	"tutup rapat pintu"		V	
11.	"buka gerbang depan"	"buka gerbang depan"		V	
12.	"gerbang tolong buka"	"gerbang tolong buka"		V	
13.	"tutup cepat"	"tutup cepat"		V	
14.	"buka pintu rumah"	"buka pintu rumah"	V		
15.	"tutup sekarang"	"tutup sekarang"		V	
16.	"buka pelan-pelan"	"buka pelan-pelan"		V	
17.	"tutup aja"	"tutup aja"		V	
18.	"buka pintunya dong"	"buka pintunya dong"		V	
19.	"tutup gerbang cepat"	"tutup gerbang cepat"		V	
20.	"buka pintu gerbang"	"buka pintu gerbang"		V	
Total Error Sistem			3		

Lampiran 3. Dokumentasi Pengujian

Lampiran 4. Kode Program Pre-Processing Audio

```
from pathlib import Path
import numpy as np
import librosa
import soundfile as sf
import noisereduce as nr
from scipy.signal import butter, filtfilt
from tqdm import tqdm

SOURCE_BASE = Path(r"D:\Skripsi\SC2\sgd-backend\sdg-bakcend\dataset")
OUTPUT_BASE = Path(r"D:\Skripsi\SC2\sgd-backend\sdg-
bakcend\dataset_processed3")

SR = 16000
FIXED_DURATION = 5.0

PRE_EMPH_COEF = 0.97
NR_PROP_DECREASE = 0.6
TARGET_RMS = 0.05

BP_LOW_HZ = 50
BP_HIGH_HZ = 7900
BP_ORDER = 4

def ensure_dir(path: Path):
    path.mkdir(parents=True, exist_ok=True)

def load_wav(path: Path, sr=SR):
    y, _ = librosa.load(path, sr=sr, mono=True)
    return y.astype(np.float32), sr

def save_wav(path: Path, y, sr=SR):
    sf.write(str(path), np.clip(y, -1.0, 1.0), sr, subtype="PCM_16")

def bandpass_filter(y, sr=SR, low=BP_LOW_HZ, high=BP_HIGH_HZ,
order=BP_ORDER):
    nyq = sr * 0.5
    b, a = butter(order, [low / nyq, high / nyq], btype="band")
    return filtfilt(b, a, y).astype(np.float32)

def pre_emphasis(y, coef=PRE_EMPH_COEF):
    return np.append(y[0], y[1:] - coef * y[:-1]).astype(np.float32)

def rms_normalize(y, target_rms=TARGET_RMS):
    rms = np.sqrt(np.mean(y ** 2) + 1e-12)
    return np.clip(y * (target_rms / rms), -1.0,
1.0).astype(np.float32)

def smart_denoise(y, sr=SR):
    if len(y) < int(sr * 0.25):
        return y
    noise_clip = y[:int(sr * 0.25)]
    try:
        return nr.reduce_noise(
            y=y, sr=sr, y_noise=noise_clip,
            stationary=True, prop_decrease=NR_PROP_DECREASE
        ).astype(np.float32)
    except Exception:
        return y
```

```
def pad_to_fixed(y, sr=SR, target_dur=FIXED_DURATION):
    target_len = int(target_dur * sr)
    if len(y) >= target_len:
        return y[:target_len]
    return np.concatenate([y, np.zeros(target_len - len(y),
dtype=np.float32)])

def process_audio(fp: Path, outdir: Path):
    ensure_dir(outdir)

    y, _ = load_wav(fp)

    if len(y) < SR * 0.3:
        print(f"⚠️ Skip (too short): {fp.name}")
        return

    y = bandpass_filter(y)
    y = pre_emphasis(y)
    y = smart_denoise(y)
    y = rms_normalize(y)
    y = pad_to_fixed(y)

    save_wav(outdir / "processed.wav", y)

def main():
    files = list(SOURCE_BASE.rglob("*.wav"))
    print(f"🎧 Total files: {len(files)}")

    for file in tqdm(files, desc="Processing dataset"):
        rel = file.relative_to(SOURCE_BASE).parts[:-1]
        out_dir = OUTPUT_BASE.joinpath(*rel) / file.stem
        process_audio(file, out_dir)

    print("\n✨ DONE — Dataset berhasil diproses konsisten (NO VAD).")


if __name__ == "__main__":
    main()
```

Lampiran 5. Kode Program Ekstraksi MFCC

```
import librosa
import numpy as np
import pandas as pd
from pathlib import Path
from tqdm import tqdm

SR = 16000
N_MFCC = 13

DATASET_PROCESSED = Path(
    r"D:\Skripsi\SC2\sgd-backend\sdg-bakcend\dataset_processed3"
)
OUTPUT_CSV = Path(
    r"D:\Skripsi\SC2\sgd-backend\sdg-
bakcend\infrastructure\storage\result\Features\mfcc_extract_processed3.
csv"
)

def extract_mfcc_52(y, sr=SR):
    rms = np.sqrt(np.mean(y ** 2) + 1e-12)
    if rms > 0:
        y = y / rms
    mfcc = librosa.feature.mfcc(
        y=y, sr=sr, n_mfcc=N_MFCC,
        n_fft=400, hop_length=160,
        fmin=50, fmax=7900
    )
    mfcc = (mfcc - np.mean(mfcc, axis=1, keepdims=True)) / (
        np.std(mfcc, axis=1, keepdims=True) + 1e-8
    )
    feats = np.concatenate([
        np.mean(mfcc, axis=1),
        np.std(mfcc, axis=1),
        np.quantile(mfcc, 0.10, axis=1),
        np.quantile(mfcc, 0.90, axis=1),
    ])
    return feats
def process_dataset():
    rows = []
    files = list(DATASET_PROCESSED.rglob("processed.wav"))
    for fp in tqdm(files, desc="Extracting MFCC"):
        y, sr = librosa.load(fp, sr=SR)
        feats = extract_mfcc_52(y)
        speaker = fp.parts[-4].lower()
        cmd = fp.parts[-3].lower()
        rows.append({
            "file_path": str(fp),
            "label_speaker": speaker,
            "label_cmd": cmd,
            **{f"f_{i}": feats[i] for i in range(52)}
        })
    df = pd.DataFrame(rows)
    df.to_csv(OUTPUT_CSV, index=False)


if __name__ == "__main__":
    process_dataset()
```

Lampiran 6. Kode Program Pre-Processing Data

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from pathlib import Path

CSV_PATH = Path(
    r"D:\Skripsi\SC2\sgd-backend\sdg-
bakcend\infrastructure\storage\result\Features\mfcc_extract_mfcc13.csv"
)

def clean_dataset(csv_path: Path):
    df = pd.read_csv(csv_path)
    df["label_speaker"] =
df["label_speaker"].astype(str).str.strip().str.lower()
    df = df.dropna()
    df = df[df["label_speaker"].isin(["me", "notme"])]
    df = df.reset_index(drop=True)
    return df

def normalize_features(df: pd.DataFrame):
    exclude = ["file_path", "label_speaker", "label_cmd"]
    feat_cols = [c for c in df.columns if c not in exclude]

    X = df[feat_cols].values
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    df_scaled = pd.DataFrame(X_scaled, columns=feat_cols)
    df_scaled["label"] = df["label_speaker"].map({"me": 1, "notme":
0}).astype(int)
    return df_scaled, scaler

if __name__ == "__main__":
    df_clean = clean_dataset(CSV_PATH)
    df_scaled, scaler = normalize_features(df_clean)

    OUT_PATH = CSV_PATH.parent / "mfcc_preprocessed2.csv"
    df_scaled.to_csv(OUT_PATH, index=False)

    print("Preprocessing selesai.")
```

Lampiran 7. Kode Program Training Model SVM

```
import pandas as pd
import numpy as np
import joblib
import time
from pathlib import Path
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, roc_auc_score,
roc_curve

DATASET = Path(
    r"D:\Skripsi\SC2\sgd-backend\sdg-
bakcend\infrastructure\storage\result\Features\mfcc_extract_mfcc13.csv"
)
MODEL_OUT = Path(
    r"D:\Skripsi\SC2\sgd-backend\sdg-
bakcend\infrastructure\storage\result\Model\svm_me_notme_mfcc13_1.jobli
b"
)

TEST_SIZE = 0.2
RANDOM_STATE = 42
TARGET_FAR = 0.05

BEST_C = 10
BEST_GAMMA = 0.1

def train_svm():
    start = time.time()

    df = pd.read_csv(DATASET)
    df["label_speaker"] = df["label_speaker"].map({"me": 1, "notme":
0})
    y = df["label_speaker"].values

    feat_cols = [c for c in df.columns if c.startswith("f_")]
    X = df[feat_cols].values

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, stratify=y, test_size=TEST_SIZE,
random_state=RANDOM_STATE
    )

    model = SVC(
        C=BEST_C,
        gamma=BEST_GAMMA,
        kernel="rbf",
        probability=True,
        random_state=RANDOM_STATE
    )
    model.fit(X_train, y_train)

    y_proba = model.predict_proba(X_test)[:, 1]
    y_pred = model.predict(X_test)

    print(classification_report(y_test, y_pred, digits=4))
    print("ROC-AUC:", roc_auc_score(y_test, y_proba))

    fpr, tpr, thr = roc_curve(y_test, y_proba)
    idx = np.argmin(abs(fpr - TARGET_FAR))
```

```
    base_thr = float(thr[idx])

    t_low = base_thr - 0.40
    t_high = base_thr - 0.16

    bundle = {
        "model": model,
        "threshold": base_thr,
        "threshold_safe_low": t_low,
        "threshold_safe_high": t_high,
        "feature_columns": feat_cols,
        "best_params": {"C": BEST_C, "gamma": BEST_GAMMA}
    }

    MODEL_OUT.parent.mkdir(parents=True, exist_ok=True)
    joblib.dump(bundle, MODEL_OUT)

    print("Model saved:", MODEL_OUT)
    print("Training time:", (time.time() - start)/60, "minutes")


if __name__ == "__main__":
    train_svm()
```

Lampiran 8. Kode Program Implementasi Regex

```
import os
import re
import tempfile
from contextlib import suppress
from pathlib import Path
from typing import Tuple, List, Dict, Optional

from pydub import AudioSegment
import speech_recognition as sr

COMMAND_PATTERNS = {
    "buka": re.compile(r"\b(buka|open)\b", re.IGNORECASE),
    "tutup": re.compile(r"\b(tutup|close)\b", re.IGNORECASE),
}


def transcribe_audio(file_path: str | Path) -> str:
    recognizer = sr.Recognizer()

    try:
        # Konversi ke format WAV 16kHz mono
        audio = AudioSegment.from_file(file_path)
        audio = audio.set_channels(1).set_frame_rate(16000)

        with tempfile.NamedTemporaryFile(delete=False, suffix=".wav")
as tmp_wav:
            audio.export(tmp_wav.name, format="wav", parameters=["-
acodec", "pcm_s16le"])
            wav_path = tmp_wav.name

        with sr.AudioFile(wav_path) as src:
            audio_data = recognizer.record(src)

        text = recognizer.recognize_google(audio_data, language="id-
ID").lower()

        return text

    except sr.UnknownValueError:
        return ""

    except sr.RequestError as e:
        print(f"Gagal menghubungi API Speech: {e}")
        return ""

    except Exception as e:
        print(f"Error transkripsi: {e}")
        return ""

    finally:
        with suppress(FileNotFoundError):
            os.remove(wav_path)


def regex_decide(text: str) -> Tuple[str, List[Dict[str, str]],
Optional[str]]:
    if not text or not text.strip():
        return "filler", [], None

    matches = []
```

```
    for cmd, pattern in COMMAND_PATTERNS.items():
        for m in pattern.finditer(text):
            matches.append({
                "command": cmd,
                "word": m.group(1).lower(),
                "span": (m.start(), m.end())
            })

    matches.sort(key=lambda x: x["span"][0])

    chosen = matches[0]["command"] if matches else None
    command = chosen if chosen in ("buka", "tutup") else "filler"

    return command, matches, chosen
```

Lampiran 9. Kode Program Integrasi MQTT

```
import os
import json
import time
import paho.mqtt.client as mqtt
from dotenv import load_dotenv
from typing import Dict

load_dotenv()
MQTT_BROKER = os.getenv("MQTT_BROKER")
MQTT_PORT = int(os.getenv("MQTT_PORT", "8883"))
MQTT_TOPIC_COMMAND = os.getenv("MQTT_TOPIC_COMMAND", "garasi/perintah")
MQTT_TOPIC_STATUS = os.getenv("MQTT_TOPIC_STATUS", "garasi/status/ml")
MQTT_USERNAME = os.getenv("MQTT_USERNAME")
MQTT_PASSWORD = os.getenv("MQTT_PASSWORD")

if not all([MQTT_BROKER, MQTT_USERNAME, MQTT_PASSWORD]):
    raise EnvironmentError("Variabel lingkungan MQTT belum lengkap.
Pastikan .env berisi konfigurasi yang benar.")

def _mk_mqtt_client() -> mqtt.Client:
    client = mqtt.Client(client_id=f"ml-{int(time.time())}",
protocol=mqtt.MQTTv311)
    client.username_pw_set(MQTT_USERNAME, MQTT_PASSWORD)
    client.tls_set()  # TLS aktif (karena port 8883)
    client.will_set(MQTT_TOPIC_STATUS, payload="offline", qos=1,
retain=True)

    client.connect(MQTT_BROKER, MQTT_PORT, keepalive=30)
    client.loop_start()
    client.publish(MQTT_TOPIC_STATUS, "online", qos=1, retain=True)
    return client

def publish_command(payload: Dict) -> bool:
    client = None
    try:
        client = _mk_mqtt_client()
        msg = json.dumps(payload, ensure_ascii=False)

        info = client.publish(MQTT_TOPIC_COMMAND, msg, qos=1,
retain=False)
        info.wait_for_publish()

    except Exception as e:
        return False

    finally:
        if client:
            time.sleep(0.2)
            client.loop_stop()
            client.disconnect()


if __name__ == "__main__":
    sample_payload = {
        "isMe": True,
        "command": "buka",
        "decision": True,
        "proba": {"Me": 0.91, "NotMe": 0.09},
    }
    publish_command(sample_payload)
```

Lampiran 10. Kode Program Implementasi Mikrokontroler (ESP32)

```
#include <ESP32Servo.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>

const char* WIFI_SSID       = "____";
const char* WIFI_PASSWORD   = "____";

const char* MQTT_BROKER     = "____";
const int   MQTT_PORT       = ____;
const char* MQTT_USER       = "____";
const char* MQTT_PASS       = "____";

const char* TOPIC_CMD       = "garasi/perintah";
const char* TOPIC_STATUS    = "garasi/status";
const char* TOPIC_STATUS_ESP= "garasi/status/esp";

const int SERVO_PIN         = 18;
const int SERVO_STOP_US     = 1500;
const int SERVO_OPEN_US     = 2200;
const int SERVO_CLOSE_US    = 800;
const unsigned long RUN_MS  = 16000;

const int IR_PIN            = 34;
const bool IR_ACTIVE_LOW    = true;
const unsigned long IR_DEBOUNCE_MS = 120;

bool irLastState            = false;
unsigned long irLastChangeMs= 0;

enum State   { STOPPED, RUN_OPEN, RUN_CLOSE };
enum DoorPos { UNKNOWN, OPEN, CLOSED };

State   currentState     = STOPPED;
DoorPos doorPos          = UNKNOWN;
DoorPos desiredDoorPos   = UNKNOWN;

unsigned long stateStartMs        = 0;
unsigned long stateBudgetMs       = 0;
unsigned long pausedRemainingMs   = 0;
bool waitingForClear              = false;

const int LED1_PIN        = 25;
const int LED2_PIN        = 26;

const unsigned long LED_BLINK_MS = 300;
bool ledState             = false;
unsigned long lastBlinkMs= 0;

const int BUZZER_PIN      = 27;

Servo servoMotor;
WiFiClientSecure tls;
PubSubClient mqtt(tls);

void updateLeds() {
  unsigned long now = millis();

  if (currentState == RUN_OPEN || currentState == RUN_CLOSE) {
```

```
    if (now - lastBlinkMs >= LED_BLINK_MS) {
      lastBlinkMs = now;
      ledState = !ledState;
      digitalWrite(LED1_PIN, ledState);
      digitalWrite(LED2_PIN, ledState);
    }
  } else {
    digitalWrite(LED1_PIN, HIGH);
    digitalWrite(LED2_PIN, HIGH);
  }
}

void servoStop()  { servoMotor.writeMicroseconds(SERVO_STOP_US); }
void servoOpen()  { servoMotor.writeMicroseconds(SERVO_CLOSE_US); }
void servoClose() { servoMotor.writeMicroseconds(SERVO_OPEN_US); }

void enterState(State s, const char* reason = nullptr) {
  currentState  = s;
  stateStartMs  = millis();
  stateBudgetMs = RUN_MS;

  switch (s) {
    case STOPPED:   servoStop();  break;
    case RUN_OPEN:  servoOpen();  desiredDoorPos = OPEN;   break;
    case RUN_CLOSE: servoClose(); desiredDoorPos = CLOSED; break;
  }

  if (reason) Serial.printf("[STATE] %s\n", reason);
}

void finishRunAndStop() {
  if (desiredDoorPos != UNKNOWN) doorPos = desiredDoorPos;
  enterState(STOPPED, "auto_stop");
}

bool isIrBlocked() {
  bool raw = digitalRead(IR_PIN);
  bool blocked = IR_ACTIVE_LOW ? (raw == LOW) : (raw == HIGH);

  if (blocked != irLastState) {
    irLastState = blocked;
    irLastChangeMs = millis();
  }

  bool stable = (millis() - irLastChangeMs) > IR_DEBOUNCE_MS;
  return stable ? irLastState : !irLastState;
}

const char* doorPosStr(DoorPos p) {
  switch (p) {
    case OPEN: return "OPEN";
    case CLOSED: return "CLOSED";
    default: return "UNKNOWN";
  }
}

const char* stateStr(State s) {
  switch (s) {
    case STOPPED: return "STOPPED";
    case RUN_OPEN: return "RUN_OPEN";
    case RUN_CLOSE: return "RUN_CLOSE";
    default: return "?";
  }
```

```
}

void publishStatus(const char* reason=nullptr, bool obstacle=false) {
  DynamicJsonDocument doc(256);

  doc["state"]     = stateStr(currentState);
  doc["door_pos"]  = doorPosStr(doorPos);
  doc["busy"]      = (currentState != STOPPED);
  doc["obstacle"]  = obstacle;
  if (reason) doc["reason"] = reason;

  char buf[256];
  size_t n = serializeJson(doc, buf);
  mqtt.publish(TOPIC_STATUS, buf, n);
}

void onMqtt(char* topic, byte* payload, unsigned int length) {
  DynamicJsonDocument doc(1024);
  if (deserializeJson(doc, payload, length)) return;

  bool isMe = doc["prediction"]["isMe"] | false;
  String cmd = String((const char*)doc["prediction"]["command"]);
  cmd.toLowerCase();

  if (!isMe) { publishStatus("rejected_notme"); return; }
  if (cmd != "buka" && cmd != "tutup") { publishStatus("invalid_cmd");
return; }
  if (cmd == "buka" && doorPos == OPEN)     {
publishStatus("already_open"); return; }
  if (cmd == "tutup" && doorPos == CLOSED)  {
publishStatus("already_closed"); return; }

  if (cmd == "buka")  enterState(RUN_OPEN, "cmd_buka");
  if (cmd == "tutup") enterState(RUN_CLOSE, "cmd_tutup");

  publishStatus(cmd.c_str());
}

void setupWifi() {
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) delay(300);
}

void ensureMqtt() {
  if (mqtt.connected()) return;

  while (!mqtt.connected()) {
    String cid = "ESP32-" + String((uint32_t)ESP.getEfuseMac(), HEX);

    if (mqtt.connect(cid.c_str(), MQTT_USER, MQTT_PASS)) {
      mqtt.subscribe(TOPIC_CMD);
      publishStatus("online");
    } else {
      delay(5000);
    }
  }
}

void tryProgress() {
  bool blocked = isIrBlocked();

  if (currentState == RUN_CLOSE && blocked) {
    unsigned long elapsed = millis() - stateStartMs;
```

```
    pausedRemainingMs = (elapsed >= stateBudgetMs) ? 1 : stateBudgetMs
- elapsed;
    waitingForClear = true;

    servoStop();
    digitalWrite(BUZZER_PIN, HIGH);

    publishStatus("paused_obstacle", true);
    enterState(STOPPED);
    return;
  }

  if (waitingForClear && !blocked) {
    waitingForClear = false;
    digitalWrite(BUZZER_PIN, LOW);

    enterState(RUN_CLOSE, "resume_after_obstacle");
    stateBudgetMs = pausedRemainingMs;
    pausedRemainingMs = 0;

    publishStatus("resumed_after_clear");
    return;
  }

  if ((currentState == RUN_OPEN || currentState == RUN_CLOSE) &&
      millis() - stateStartMs >= stateBudgetMs) {
    finishRunAndStop();
  }
}

void setup() {
  Serial.begin(115200);

  pinMode(LED1_PIN, OUTPUT);
  pinMode(LED2_PIN, OUTPUT);

  pinMode(IR_PIN, INPUT_PULLUP);
  irLastState = isIrBlocked();

  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(BUZZER_PIN, LOW);

  servoMotor.attach(SERVO_PIN, 500, 2500);
  servoStop();

  tls.setInsecure();
  mqtt.setServer(MQTT_BROKER, MQTT_PORT);
  mqtt.setCallback(onMqtt);
  mqtt.setBufferSize(1024);

  setupWifi();

  currentState = STOPPED;
  publishStatus("boot_idle");
}

void loop() {
  ensureMqtt();
  mqtt.loop();
  tryProgress();
  updateLeds();
}
```